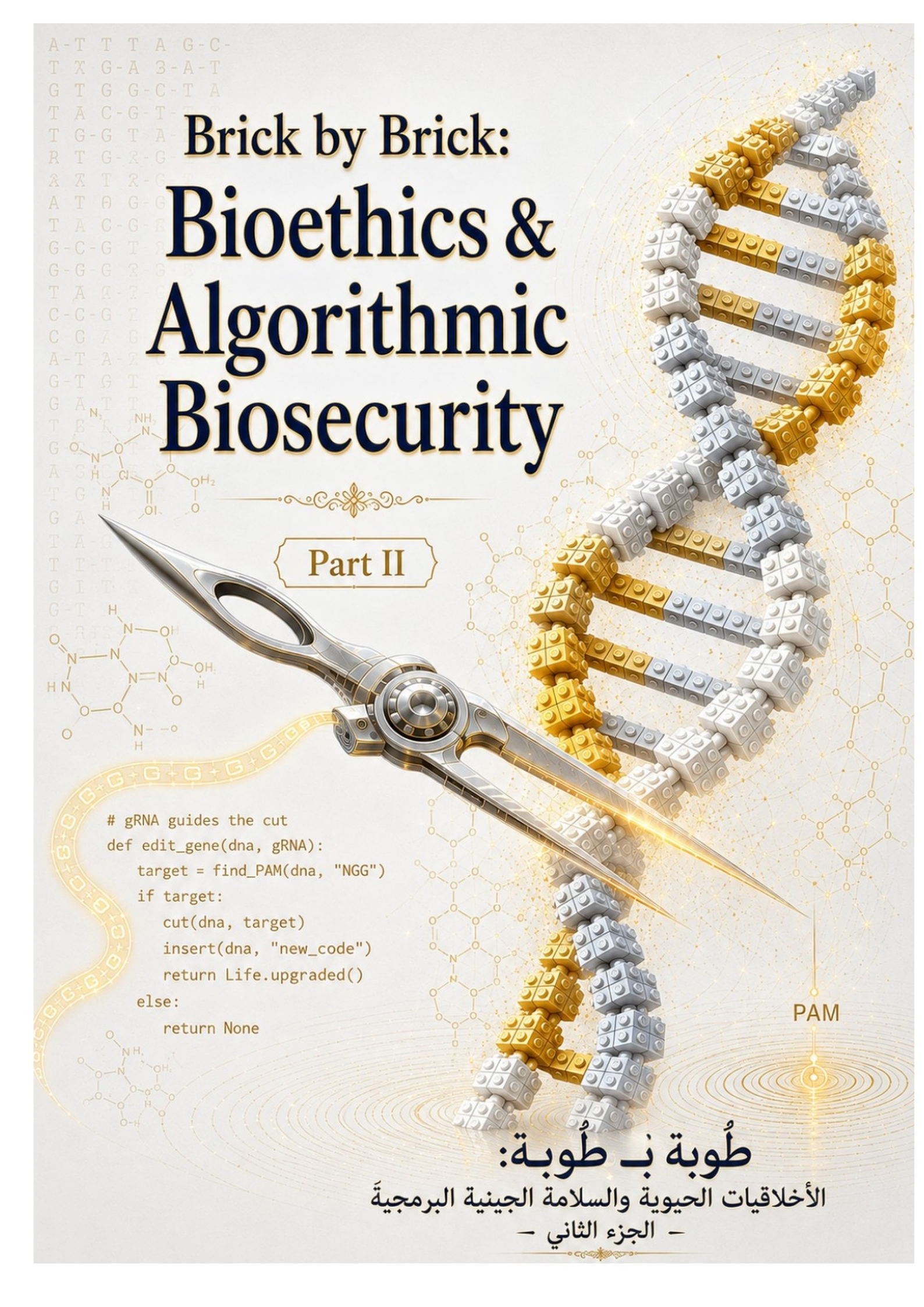
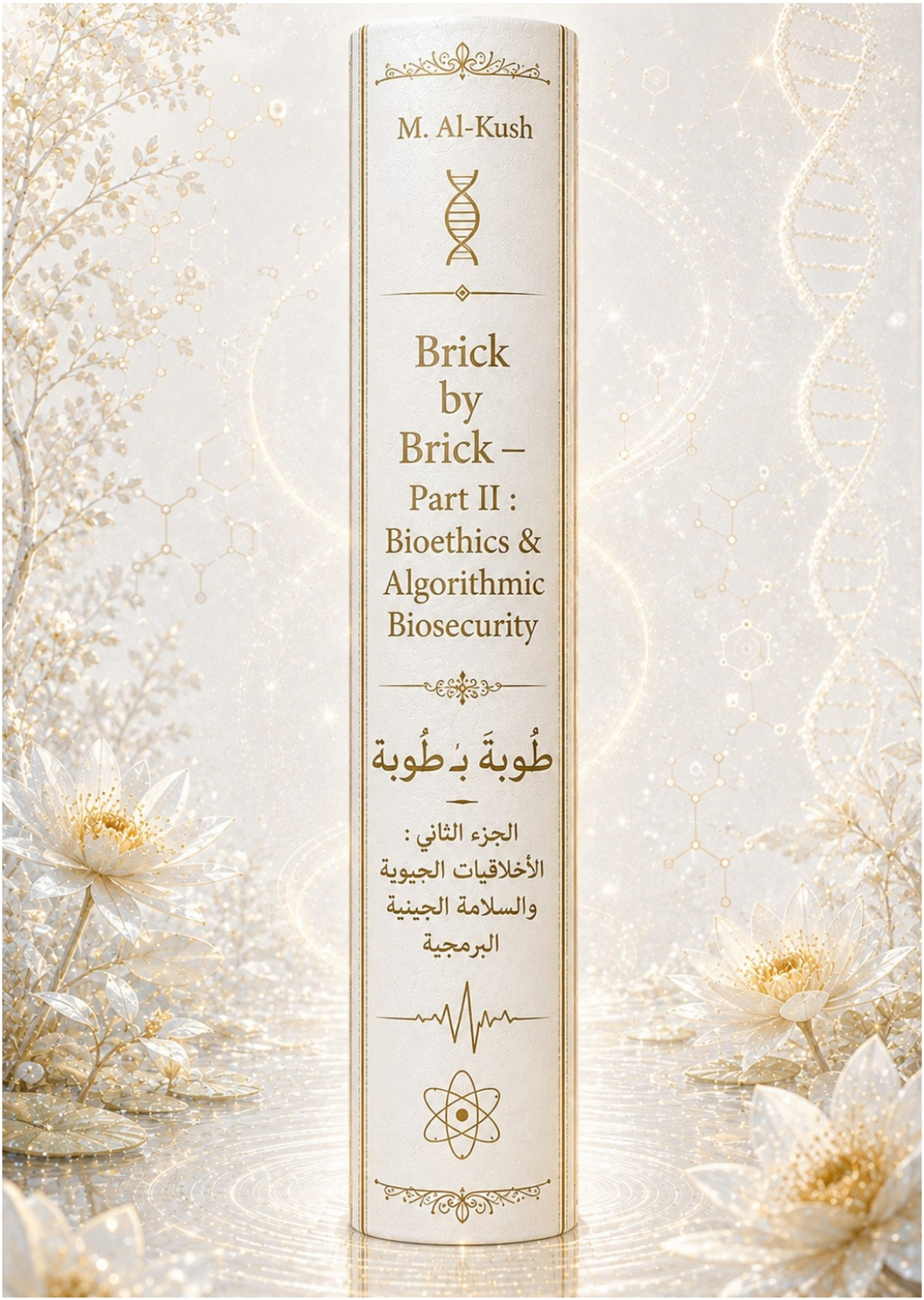
Brick by Brick:
Bioethics & Algorithmic Biosecurity

Part II




```
# gRNA guides the cut
def edit_gene(dna, gRNA):
    target = find_PAM(dna, "NGG")
    if target:
        cut(dna, target)
        insert(dna, "new_code")
        return Life.upgraded()
    else:
        return None
```

طُوبة بـ طُوبـة:
الأخلاقيات الحيوية والسلامة الجينية البرمجيةَ
– الجزء الثاني –


M. Al-Kush
Brick by Brick –
Part II : Bioethics & Algorithmic Biosecurity
طُوبةَ بـِ طُوبة
الجزء الثاني :
الأخلاقيات الجيوية والسلامة الجينية البرمجية

طوبةٌ بـ طُوبة

الأخلاقيات الحيوية والسلامة الجينية البرمجية

— الجزء الثاني —

Brick by Brick
Building the Future, One Code at a Time.

9 781234 567897

Brick by Brick: Bioethics & Algorithmic Biosecurity

طوبة بـ طوبة: الأخلاقيات الحيوية والسلامة الجينية البرمجية

Part II

الجزء الثاني

حقوق الملكية والترخيص

Copyright and Licensing

© 2026 م. الزغبي عنتر الكوش. جميع الحقوق محفوظة. البريد الإلكتروني:

.mohamedelzoghpe905@gmail.com

© 2026 M. El-Zoghpy Antar Al-Kush. All rights reserved.
Email: mohamedelzoghpe905@gmail.com.

المفاهيم العلمية والمراجع العلمية الواردة في هذا العمل مستمدة من مصادر أكاديمية متاحة للعامة ومفتوحة الوصول كما هي مدرجة في قائمة المراجع. أما النص العربي الأصلي، والتبسيطات التعليمية، والاستعارات البصرية، والرسوم المعلوماتية، وتصميمات الفن الرقمي، والرسوم التوضيحية، والترجمات الإنجليزية، والتجميع المنهجي العام المقدم في هذا الكتاب فهي من الملكية الفكرية الحصرية للمؤلف/المصمم ومحمية بموجب قوانين حقوق النشر الدولية.

The scientific concepts and references cited in this work are derived from public domain and open-access academic sources as listed in the bibliography. However, the original Arabic text, educational simplifications, visual metaphors, infographics, digital art designs, illustrations, English translations, and the overall methodological compilation presented in this book are the exclusive intellectual property of the author/designer and are protected under international copyright laws.

هذا العمل مرخص بموجب رخصة المشاع الإبداعي: نسب المصنف - غير تجاري 4.0 دولي (CC BY-NC 4.0).

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Table of Contents

فهرس المحتويات

English	Page	العربية
Preliminary Chapter — Foundation Stage	11	الفصل التمهيدي — مرحلة التأسيس
Part 1 — Introduction to Bioinformatics	22	الباب 1 — المدخل إلى المعلوماتية الحيوية
Chapter 1 — What Is Bioinformatics	22	الفصل 1 — ما المعلوماتية الحيوية؟
Chapter 2 — From DNA to a Data File	31	الفصل 2 — من DNA إلى ملف بيانات
Chapter 3 — Data Quality and Reproducibility	40	الفصل 3 — جودة البيانات وإمكانية إعادة الإنتاج
Part 2 — Foundation: Preparing the Working Environment	48	الباب 2 — مرحلة التأسيس: تجهيز بيئة العمل
Chapter 4 — Installing Python and Understanding PATH	48	الفصل 4 — تثبيت Python وفهم PATH
Chapter 5 — Virtual Environments and Packages	58	الفصل 5 — البيئات الافتراضية والحزم
Chapter 6 — First Encounter with Biopython	66	الفصل 6 — أول لقاء مع Biopython
Part 3 — Python as the Language Foundation for Computational Biology	75	الباب 3 — Python كأساس لغوي للبيولوجيا الحاسوبية
Chapter 7 — Data Types	75	الفصل 7 — أنواع البيانات
Chapter 8 — Functions, Modules, and Error Handling	82	الفصل 8 — الدوال والوحدات وإدارة الأخطاء
Chapter 9 — Object-Oriented Programming	91	الفصل 9 — البرمجة الكائنية
Part 4 — The Practical Core: Biopython	104	الباب 4 — القلب النابض: Biopython
Chapter 10 — Seq: Sequence as an Object	104	الفصل 10 — Seq: التسلسل ككائن
Chapter 11 — SeqRecord: Sequence with Identity	112	الفصل 11 — SeqRecord: التسلسل مع هويته

Chapter 12 — SeqFeature: Representing Locations and Features	121	الفصل 12 — SeqFeature: تمثيل المواقع والميزات
Chapter 13 — SeqIO: Reading and Writing Files	130	الفصل 13 — SeqIO: قراءة وكتابة الملفات
Chapter 14 — FASTA, FASTQ, and GenBank	138	الفصل 14 — FASTA و FASTQ و GenBank
Chapter 15 — Alignment, Similarity, and Homology	152	الفصل 15 — المحاذاة والتشابه والتماثل
Part 5 — Supporting Tools	159	الباب 5 — الأدوات المساندة
Chapter 16 — Bash and the Command Line	160	الفصل 16 — Bash وسطر الأوامر
Chapter 17 — Perl in a Limited Dose	168	الفصل 17 — Perl بجرعة محددة
Chapter 18 — R for Statistics and Visualization	177	الفصل 18 — R والإحصاء والتصور
Part 6 — Molecular Biology for the Programmer	198	الباب 6 — البيولوجيا الجزيئية التي يحتاجها المبرمج
Chapter 19 — Bioconductor	198	الفصل 19 — Bioconductor
Chapter 20 — DNA, RNA, and Protein	207	الفصل 20 — DNA و RNA والبروتين
Chapter 21 — Genes and Gene Expression	216	الفصل 21 — الجين والتعبير الجيني
Chapter 22 — Mutations and Genetic Variation	226	الفصل 22 — الطفرات والاختلاف الوراثي
Chapter 23 — Regulation, Expression, and Response	235	الفصل 23 — التنظيم والتعبير والاستجابة
Part 7 — Biological Algorithms	245	الباب 7 — الخوارزميات الحيوية
Chapter 24 — Sequence Search	245	الفصل 24 — البحث في التسلسل
Chapter 25 — Hamming and Levenshtein Distance	252	الفصل 25 — Hamming و Levenshtein
Chapter 26 — Computational Complexity	261	الفصل 26 — التعقيد الحسابي
Part 8 — CRISPR from a Scientific and Secure Computational Perspective	272	الباب 8 — CRISPR من منظور علمي وحاسوبي آمن

Chapter 27 — Clustering	272	الفصل 27 — Clustering
Chapter 28 — CRISPR-Cas9 History and Conceptual Mechanism	283	الفصل 28 — تاريخ CRISPR-Cas9 ومبدأه
Chapter 29 — Representing Features with SeqFeature	291	الفصل 29 — تمثيل المواقع باستخدام SeqFeature
Chapter 30 — Off-Target Analysis as a Safety Function	299	الفصل 30 — Off-target analysis كوظيفة وقائية
Part 9 — Bioethics and Algorithmic Biosecurity	309	الباب 9 — الأخلاقيات الحيوية والسلامة الجينية البرمجية
Chapter 31 — Bioethics, Biosafety, and Biosecurity	309	الفصل 31 — Bioethics و Biosafety و Biosecurity
Chapter 32 — Sequence Screening	317	الفصل 32 — Sequence Screening
Chapter 33 — Why Sequence Matching Is Not Enough	327	الفصل 33 — لماذا لا تكفي مطابقة التسلسل؟
Part 10 — The biobrick_security Educational Package	335	الباب 10 — مشروع biobrick_security كحزمة تعليمية
Chapter 34 — Gene Drive Constraints	335	الفصل 34 — Gene Drive Constraints
Chapter 35 — DNA Watermarking	344	الفصل 35 — DNA Watermarking
Chapter 36 — Biosafety Codon Filtering	355	الفصل 36 — Biosafety Codon Filtering
Chapter 37 — Engineering the Educational Security Package	365	الفصل 37 — هندسة الحزمة الأمنية التعليمية
Chapter 38 — pathogen_check	375	الفصل 38 — pathogen_check
Chapter 39 — off_target_scan	386	الفصل 39 — off_target_scan
Chapter 40 — gene_drive_lock	396	الفصل 40 — gene_drive_lock
Chapter 41 — Should the Package Be Published on PyPI?	408	الفصل 41 — هل تُنشر الحزمة على PyPI؟
Part 11 — RNA-seq: From Sequence to Gene Expression	417	الباب 11 — RNA-seq: من التسلسل إلى التعبير الجيني

Chapter 42 — What Does RNA-seq Measure?	417	الفصل 42 — ما الذي يقيسه RNA-seq؟
Chapter 43 — Read Quality	427	الفصل 43 — جودة القراءات
Chapter 44 — Counts and Normalization	438	الفصل 44 — Counts والتطبيع
Chapter 45 — Differential Expression	450	الفصل 45 — التعبير التفاضلي
Chapter 46 — PCA and Clustering	461	الفصل 46 — PCA و Clustering
Chapter 47 — The Bridge Between RNA-seq and Biosecurity	472	الفصل 47 — الجسر بين RNA-seq والأمن الحيوي
Part 12 — Biostatistics	486	الباب 12 — الإحصاء الحيوي
Chapter 48 — Description Before Inference	486	الفصل 48 — الوصف قبل الاستدلال
Chapter 49 — Probability and Distributions	497	الفصل 49 — الاحتمال والتوزيعات
Chapter 50 — Hypothesis Testing	507	الفصل 50 — اختبار الفرضيات
Chapter 51 — Multiple Comparisons	517	الفصل 51 — المقارنات المتعددة
Part 13 — Database Management and Scientific Resources	527	الباب 13 — إدارة قواعد البيانات والمصادر العلمية
Chapter 52 — Searching PubMed and Biological Databases	528	الفصل 52 — البحث في PubMed وقواعد البيانات
Chapter 53 — Local Datasets versus Live Databases	538	الفصل 53 — البيانات المحلية مقابل البيانات الحية
Chapter 54 — Documentation and Reproducibility	547	الفصل 54 — التوثيق وقابلية إعادة الإنتاج
Part 14 — Integrated Capstone Project	560	الباب 14 — مشروع التخرج المتكامل
Chapter 55 — The Integrated Brick by Brick Capstone	560	الفصل 55 — مشروع Brick by Brick
Chapter 56 — The Scientific Report	590	الفصل 56 — التقرير العلمي
Part 15 — Advanced Practice: Real Data, Statistical Models, Workflows, and AI-Assisted	593	الباب 15 — الممارسة المتقدمة:

Biology		البيانات الحقيقية، النماذج الإحصائية، إدارة سير العمل، والذكاء الاصطناعي في البيولوجيا
Chapter 57 — From Clean Exercises to Real Sequencing Data	594	الفصل 57 — من التمارين النظيفة إلى بيانات التسلسل الحقيقية
Chapter 58 — RNA-seq Statistics Beyond the t-test	599	الفصل 58 — إحصاء RNA-seq بعد اختبار t
Chapter 59 — Real Reproducible Workflows: Snakemake, Nextflow, Conda, Docker, and Apptainer	605	الفصل 59 — سير العمل القابل لإعادة الإنتاج: Snakemake وNextflow وConda وDocker وApptainer
Chapter 60 — From Educational Biosecurity to Operational Defensive Engineering	610	الفصل 60 — من نموذج الأمن الحيوي التعليمي إلى الهندسة الدفاعية التشغيلية
Chapter 61 — Generative AI in Daily Bioinformatics Practice	613	الفصل 61 — الذكاء الاصطناعي التوليدي في الممارسة اليومية للمعلوماتية الحيوية
Chapter 62 — The Real-Data Capstone: From FASTQ to Defensible Conclusion	617	الفصل 62 — المشروع التطبيقي بالبيانات الحقيقية: من FASTQ إلى استنتاج يمكن الدفاع عنه
Part 16 — Deep Practice and Verification	621	الباب 16 — الممارسة العميقة والتحقق
Chapter 63 — Public Data Acquisition and Provenance	621	الفصل 63 — الحصول على البيانات العامة ومصدرها وتتبّعها
Chapter 64 — RNA-seq Diagnostics, Designs, and Effect Interpretation	625	الفصل 64 — تشخيص RNA-seq وتصميماته وتفسير التأثيرات
Chapter 65 — Workflow Engineering at Scale	628	الفصل 65 — هندسة سير العمل على نطاق واسع
Chapter 66 — AI-Assisted Analysis, Validation, and Scientific Judgment	631	الفصل 66 — التحليل بمساعدة الذكاء الاصطناعي والتحقق والحكم العلمي

Installing Python and Understanding PATH.

1. تثبيت Python وفهم PATH

لا يكفي أن يكون يوجد في المثال برنامج Python على حاسوبك؛ بل يجب أن تعرف أي نسخة منه يستجيب عندما تكتب أمراً في الطرفية. نظام التشغيل يعتمد على متغير بيئي يُدعى PATH، وهو أشبه بقائمة عناوين يبحث فيها الحاسوب عن البرامج. إذا كان يوجد في المثال أكثر من تثبيت واحد، فقد يشير الأمر python إلى نسخة قديمة أو إلى بيئة مختلفة تماماً. لذلك، أول خطوة عملية هي التحقق من الإصدار الفعلي المستجيب، وتوثيق ذلك كجزء من سجل التجربة، لأن تغير المفسر قد يغير النتيجة ذاتها.

It is not enough to have Python installed; you must know which interpreter responds when you type a command. The operating system relies on an environment variable called PATH, which functions like a list of directories where executables are searched for. If the workflow has multiple installations, the command python might point to an older version or a different environment entirely. Thus, the first practical step is to verify the active interpreter version and document it as part of the experimental record, because changing the interpreter can change the outcome itself.

Language / نوع الكود: **Bash / Shell**

الغرض: تنفيذ خطوة تشغيلية محددة من الطرفية بحيث يمكن تسجيلها وإعادة تشغيلها.

Purpose: execute a defined command-line step so it can be recorded and rerun.

```
python --version
py --version   # على نظام Windows
```

أوامر الطرفية حساسة للسياق: نظام التشغيل، PATH، مجلد العمل، صلاحيات الملفات، وإصدارات الأدوات. لذلك يجب تسجيل الأمر كما نُفذ، مع مدخلاته ومخرجاته، بدل حفظ النتيجة وحدها.

اختبار الكود يجب أن يثبت السلوك المتوقع وحالات الفشل المتوقعة، لا مجرد أن المفسر قبل الصياغة.

When files, tables, biological records, or workflow steps are involved, provenance is part of the computation rather than an administrative afterthought. A robust test checks both expected behavior and failure behavior, while the biological conclusion remains dependent on the data source, experimental design, reference version, and documented software environment.

Command-line reproducibility requires a defined working directory, environment, paths, correct quoting, and explicit inspection of exit status for important steps. The same command can behave differently across systems

when tools, permissions, or path conventions differ, so the execution context should be recorded with the command.

في الطرفية، إعادة الإنتاج تتطلب تثبيت مجلد العمل والبيئة والمسارات، واستخدام الاقتباس الصحيح للمتغيرات، وفحص رمز الخروج لكل خطوة مهمة. إعادة تشغيل الأمر على نظام مختلف قد تغير النتيجة إذا اختلفت الأدوات أو الصلاحيات أو بنية المسارات؛ لذلك ينبغي تسجيل السياق إلى جانب الأمر.

For workflow code, one successful run is not enough. Inputs, outputs, paths, and logs must be explicit, and failures should remain visible rather than being silently ignored. That distinction separates a command from an auditable, reproducible workflow.

في هذا النوع من الأكواد لا يكفي أن ينجح الأمر مرة واحدة. يجب أن تكون المدخلات والمخرجات والمسارات والسجلات واضحة، وأن يكون الفشل ظاهرًا بدل أن يمر بصمت. هذا هو الفرق بين أمر منفرد وخط عمل قابل للتدقيق وإعادة التشغيل.

On some Unix systems, the command python may not point to Python 3 by default; you may need to use python3 --version instead. This is not a technical problem, but part of documentation precision: knowing the exact interpreter name you invoke is the first step toward reproducibility. Always record the full command name you used.

Creating a Virtual Environment

على بعض أنظمة Unix، قد لا يستجيب الأمر python لإصدار Python 3 افتراضياً، بل قد تحتاج إلى استخدام python3 --version. هذه ليست مشكلة تقنية، بل هي جزء من دقة التوثيق: معرفة الاسم الدقيق للمفسر الذي تستدعيه هي أولى خطوات إعادة الإنتاج. سجل دائماً الاسم الكامل للأمر الذي استخدمته.

A virtual environment is an "isolated workspace" for the project. Instead of installing all libraries system-wide (which may cause conflicts between projects), a virtual environment allows you to install specific package versions only within the project folder. This isolation makes reproducibility feasible, because the workflow can save an exact list of the dependencies you used. Creating an environment is not a luxury; it is part of the scientific method in the digital age.

2. إنشاء البيئة الافتراضية

البيئة الافتراضية هي "غرفة عمليات معزولة" للمشروع. بدلاً من تثبيت جميع المكتبات على النظام الرئيسي (مما قد يسبب تعارضات بين المشاريع)، تسمح لك البيئة الافتراضية بتثبيت إصدارات محددة من الحزم داخل مجلد المشروع فقط. هذا العزل يجعل إعادة إنتاج النتائج أمراً ممكناً، لأنك يمكن حفظ قائمة بالاعتماديات الدقيقة التي استخدمتها. إن إنشاء البيئة ليس رفاهية، بل هو جزء من المنهج العلمي في العصر الرقمي.

الغرض: إنشاء أو تفعيل بيئة معزولة للمشروع وتوثيق الأوامر التي تبني البيئة.

Purpose: create or activate an isolated project environment and document the commands used to build it.

Language / نوع الكود: **Bash / Shell**

```
# إنشاء البيئة (على نظام Unix/macOS)
python -m venv .venv
# تفعيل البيئة (Unix/macOS)
source .venv/bin/activate
# تفعيل البيئة (Windows)
.venv\Scripts\activate
```

The activation method differs by operating system. On Windows: .venv\Scripts\activate; on Unix or macOS, use: source .venv/bin/activate. This difference is not a flaw, but a reminder that the software environment is not just code—it is an interaction with a specific operating system, and this interaction must also be documented in the reproducibility record.

وتختلف طريقة التفعيل حسب نظام التشغيل. على Windows: .venv\Scripts\activate، أما على Unix أو macOS فتُفعّل بالأمر: source .venv/bin/activate. هذا الاختلاف ليس عيباً، بل هو تذكير بأن البيئة البرمجية ليست مجرد كود، بل هي تفاعل مع نظام تشغيل محدد، وهذا التفاعل يجب توثيقه أيضاً في سجل إعادة الإنتاج.

الغرض: تثبيت أو تحديث حزمة Python أو تسجيل الاعتماديات المستخدمة في البيئة.

Purpose: install or update a Python package or record the dependencies used by the environment.

Language / نوع الكود: **Bash / Shell**

```
# تحديث أداة pip نفسها
python -m pip install --upgrade pip
```

Installing Biopython and Verifying the Installation.

Biopython is the practical core of this book. After activating the virtual environment, it is installed via pip, but what matters most is verifying the installation and reading the version number. This number must be recorded, because just as gene translation differs with the reading frame, code outputs may differ with different library versions.

3. تثبيت Biopython والتحقق من نجاحه

Biopython هو القلب العملي لهذا الكتاب. بعد تفعيل البيئة الافتراضية، يُثبت باستخدام pip، لكن الأهم هو التحقق من نجاح التثبيت وقراءة رقم الإصدارة. هذا الرقم يجب تسجيله، لأنه كما تختلف ترجمة الجين باختلاف الإطار القرائي، قد تختلف مخرجات الكود باختلاف إصدارة المكتبة.

Language / نوع الكود: **Bash** / **Shell**

Purpose: perform a defined computational step while keeping inputs and outputs traceable.

الغرض: تنفيذ خطوة حسابية محددة ضمن سير العمل مع إبقاء المدخلات والمخرجات قابلة للتتبع.

```
python -m pip install biopython
python -c "import Bio; print(Bio.__version__)"
```

Read this example in three layers: inputs, the transformation performed by functions or objects, and the resulting output or checks. Intermediate variables represent explicit states, while the final calls define what is emitted or passed to the next stage.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. المتغيرات تمثل حالات وسيطة يجب أن تبقى واضحة الاسم والنوع، بينما تحدد الاستدعاءات النهائية ما الذي سيظهر للمستخدم أو يدخل في الخطوة التالية.

After installing the required packages, use pip to freeze the list of dependencies and their exact versions into a text file (e.g., requirements.txt). This file is not merely administrative; it is the digital fingerprint of the environment, allowing any other researcher (or the analysis itself a year later) to rebuild the exact same environment literally. With this step, the environment transforms from a transient state into a fixed scientific artifact.

بعد تثبيت الحزم المطلوبة، تُستخدم أداة pip لتجميد قائمة الاعتماديات وإصداراتها في ملف نصي (مثل requirements.txt). هذا الملف ليس مجرد سجل إداري؛ بل هو البصمة الرقمية للبيئة، التي تسمح لأي باحث آخر (أو لنفسك بعد عام) بإعادة بناء نفس البيئة حرفياً. بهذه الخطوة، تتحول البيئة من حالة عابرة إلى أثر علمي ثابت.

Language / نوع الكود: **Bash** / **Shell**

```
pip freeze > requirements.txt
```

This block represents data or a command rather than Python source. Its type matters because its interpretation, validation, and copying rules differ from executable Python.

هذه الكتلة تمثل بيانات أو أمرًا لا ينبغي معاملته كبرنامج Python. نوع المحتوى مهم لأن تفسيره، واختباره، وطريقة نسخه تختلف عن الكود التنفيذي.

First Short Programming Test

This section begins with to break the ice. We pass a simple string ("ATGC") to a Seq object, then ask it to compute the complement and the reverse complement. The goal here is not to perform gene editing, but to test that the environment works, and to understand that a Seq object is not just text; it represents a biological sequence with its own direction and golden rules.

4. أول تجربة برمجية قصيرة

الآن تبدأ هنا مرحلة كسر الجمود. نمرر سلسلة نصية بسيطة ("ATGC") إلى كائن Seq، ثم نطلب منه حساب المتمم (complement) والمتمم العكسي (reverse_complement). الهدف هنا ليس إجراء تعديل جيني، بل اختبار أن البيئة تعمل، وفهم أن كائن Seq ليس مجرد نص؛ إنه يمثل تسلسلاً بيولوجياً يحمل اتجاهه الخاص وقواعده الذهبية.

Purpose: create a Seq object and test reverse-complement operations on a synthetic sequence.

الغرض: إنشاء كائن Seq واختبار عمليات التكامل العكسي على تسلسل اصطناعي.

from Bio.Seq import Seq

تسلسل اصطناعي صغير للاختبار التعليمي

dna = Seq("ATGC")

حساب المتمم

```
print(dna.complement())  # الـنـا تج: TACG
# حساب الـمـتـمم الـعـكسي
print(dna.reverse_complement()) # الـنـا تج: GCAT
```

In a real deployment, the Python version and package set should be pinned, inputs should be validated before execution, and failures and outputs should be recorded. File-oriented functions should define behavior for missing, empty, malformed, or unexpectedly encoded inputs. When code represents a biological rule, tests should include valid, boundary, and rejected cases, with the test result recorded alongside the code version.

في التطبيق الواقعي ينبغي تثبيت نسخة Python والحزم، والتحقق من أنواع المدخلات قبل تشغيل المسار، وتسجيل الاستثناءات والمخرجات. إذا كانت الدالة تتعامل مع ملف، فينبغي أيضًا تحديد ما يحدث عند غيابه أو فراغه أو احتوائه على ترميز غير متوقع. وعندما يمثل الكود قاعدة بيولوجية، يجب اختبار القاعدة بحالات صحيحة وحالات حدية وحالات مرفوضة، ثم حفظ نتيجة الاختبار مع إصدار الكود.

Fundamental rule: all sequences used in this chapter's exercises are purely synthetic, crafted to teach programming concepts, not to represent any real organism. This deliberate isolation ensures the code works anytime without an internet connection, and frees the mind from worrying about the biological significance of the results, allowing focus instead on the accuracy of computational representation and the integrity of transformations.

قاعدة أساسية: كل التسلسلات المستخدمة في تمارين هذا الفصل هي اصطناعية بحتة، صُنعت لتعليم المفاهيم البرمجية وليس لتمثيل أي كائن حي حقيقي. هذا العزل المتعمد يضمن أن يعمل الكود في أي وقت دون اتصال بالإنترنت، ويحرر الذهن من القلق حول الدلالة البيولوجية للنتائج، لتركز بدلاً من ذلك على دقة التمثيل البرمجي وسلامة التحويلات.

Rule for Clear Code Comments

When writing educational or research code, let comments explain the reason (the purpose of the code), not what the line does (which is obvious from the code itself). A good comment explains why the function was chosen, what hypothesis the code tests, or why certain values were filtered out. This practice transforms code from mere machine instructions into a reviewable scientific document.

5. قاعدة التعليقات البرمجية

عندما تكتب كوداً تعليمياً أو بحثياً، اجعل التعليقات تشرح السبب (الغرض من الكود)، وليس ماذا يفعل السطر (فهذا واضح من الكود نفسه). التعليق الجيد يوضح سبب اختيار هذه الدالة، أو الفرضية التي يختبرها الكود، أو سبب استبعاد قيم معينة. هذه الممارسة تحول الكود من مجرد تعليمات للآلة إلى وثيقة علمية قابلة للمراجعة.

```
#صحيح: شرح الغرض
#حساب المتمم العكسي للتحقق من اتجاهية الشريط الهدف
#خاطئ: تكرار ما يفعله السطر
#استدعاء دالة reverse_complement
```

Defensive programming begins with input validation. Before performing any operation on a sequence, ensure it contains only allowed characters (A, T, C, G). Using a function like `set(sequence).issubset('ATCG')` protects the code from crashing due to an unexpected character, turning a potential error into a clear message. This is the coding embodiment of the "fail-securely" principle before we even reach the advanced security gates.

البرمجة الدفاعية تبدأ بالتحقق من صحة المدخلات. قبل إجراء أي عملية على تسلسل، يجب التأكد من أنه يحتوي على الحروف المسموحة فقط (A، T، C، G). استخدام دالة مثل `set(sequence).issubset('ATCG')` يحمي الكود

من الانهيار بسبب حرف غير متوقع، ويحوّل الخطأ المحتمل إلى رسالة واضحة. هذا هو التجسيد البرمجي لمبدأ "فشل آمن" (Fail-securely) قبل أن نصل إلى البوابات الأمنية المتقدمة.

Purpose: turn a repeatable step into a function with explicit inputs, outputs, and testable failure states.

الغرض: تحويل خطوة متكررة إلى دالة ذات مدخلات ومخرجات وحالات فشل يمكن اختبارها.

```
def validate_dna(seq):
    if not set(seq.upper()).issubset('ATCG'):
        raise ValueError(f"التسلسل يحتوي على أحرف غير صالحة: {seq}")
    return seq.upper()
```

الدوال المعرفة هنا (validate_dna) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

Read this example in three layers: inputs, the transformation performed by functions or objects, and the resulting output or checks. The defined function separates one operation from the surrounding workflow, making it possible to test both normal and failure cases before reuse.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. الدالة/الدوال المعرّفة هنا تفصل العملية عن بقية البرنامج، ولذلك يمكن اختبارها على حالة صحيحة وحالة فشل قبل إدخالها في خط عمل أكبر.

Object-Oriented Programming (OOP) in Biopython.

6. البرمجة الكائنية (OOP) في Biopython

تعتمد Biopython على نموذج البرمجة الكائنية لتقديم تمثيل غني للكيانات البيولوجية. هناك ثلاثة مستويات أساسية:

- Seq: يمثل التسلسل نفسه (الحروف) مع عمليات بيولوجية.

Biopython relies on the object-oriented programming model to provide a rich representation of biological entities. There are three fundamental layers:

- SeqRecord: يغلف التسلسل بهوية (مثل id) ووصف وبيانات وصفية، ليكون سجلاً كاملاً.

· Seq: Represents the sequence itself (the letters) with biological operations.

· SeqFeature: يمثل ميزة موضعية على التسلسل (مثل موقع بداية الجين)، مع إحداثيات واتجاه.

· SeqRecord: Wraps the sequence with an identity (e.g., id), a description, and metadata, forming a complete record.

هذا التغليف يجعل العلاقات بين البيانات واضحة، بدلاً من تشتيتها في متغيرات منفصلة، ويمهد الطريق للتعامل مع الملفات الحقيقية لاحقاً.

· SeqFeature: Represents a localized feature on the sequence (e.g., a gene start site), with coordinates and strand information.

This encapsulation makes the relationships among data explicit, rather than scattering them across separate variables, and paves the way for handling real files later.

Representing Features with SeqFeature.

7. تمثيل المواقع باستخدام SeqFeature

Any location on DNA is defined by three essential elements: start, end, and strand (+ or -). The SeqFeature object takes these coordinates and gives them biological meaning, allowing them to be linked to a description (e.g., "synthetic target"). This precise representation protects you from common errors, such as confusing zero-based and one-based coordinates, or ignoring the strand which entirely changes the feature's interpretation.

أي موقع على الـ DNA يُعرّف بثلاثة عناصر أساسية: البداية، النهاية، والاتجاه (+ أو -). كائن SeqFeature يأخذ هذه الإحداثيات ويعطيها معنى، بحيث يمكن ربطها بوصف (مثل "هدف اصطناعي"). هذا التمثيل الدقيق يحميك من أخطاء شائعة، مثل الخلط بين الإحداثيات الصفرية والأحادية، أو تجاهل الاتجاه الذي يغير تفسير الميزة تماماً.

Purpose: represent a feature and its location while keeping coordinates explicit as computational representation.

الغرض: تمثيل ميزة وموقعها على التسلسل، مع إبقاء الإحداثيات جزءاً من التمثيل الحسابي.

from Bio.SeqFeature import SeqFeature, FeatureLocation

تمثيل موقع اصطناعي (من القاعدة 4 إلى 10 على الشريط الموجب)

feature = SeqFeature(

```
FeatureLocation(start=4, end=10, strand=+1),
type="synthetic_target",
```

qualifiers={"note": "موقع تعليمي للتمرين على الإحداثيات"}

```
)
```

```
print(feature.location.start, feature.location.end, feature.strand)
```

form the complete picture: the sequence (Seq) is loaded inside a record (SeqRecord) carrying its identity, then a feature (SeqFeature) is added to specify a particular location on it. This hierarchical composition (encapsulation) mirrors how genes are stored in real databases like GenBank, where the sequence is never stored alone, but surrounded by its description and coordinates.

لتكوين صورة كاملة: التسلسل (Seq) يُحمل داخل سجل (SeqRecord) يحمل هويته، ثم تُضاف إليه ميزة (SeqFeature) تحدد موقعاً معيناً عليه. هذا التركيب الهرمي (التغليف) يعكس الطريقة التي تُخزّن بها الجينات في قواعد البيانات الحقيقية مثل GenBank، حيث لا يُخزّن التسلسل وحيداً، بل محاطاً بوصفه وإحداثياته.

Purpose: bind a sequence to identity, description, and metadata in one record.

الغرض: ربط التسلسل بهوية ووصف وبيانات وصفية في سجل واحد.

from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio.SeqFeature import SeqFeature, FeatureLocation

إنشاء التسلسل

my_seq = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC")

إنشاء السجل مع هوية

record = SeqRecord(my_seq, id="SYN_001", description="جين اصطناعي للتمرين")

عرض السجل مع ميزته

```
print(record.id, record.description)
print(record.features[0].location)
# إضافة ميزة (من القاعدة 5 إلى 15 على الشريط الموجب)
feature = SeqFeature(FeatureLocation(5, 15, strand=+1), type="exon")
record.features.append(feature)
```

The Role of Each Language in the Workshop.

8. توزيع اللغات في الورشة

لكل أداة مكانها في ورشة المعلوماتية الحيوية. تخيل الموقف التالي:

· Bash: هو فريق تجهيز موقع البناء، يدير الملفات، ويشغل الأدوات، ويربط المدخلات بالمخرجات في خطوط أنابيب منظمة.

· Biopython (مع Python): هو المهندس الذكي الذي يفهم المخطط الجيني، يقرأ التسلسلات، يحللها، ويحولها.

· R (مع Bioconductor): هو المفتش الإحصائي الذي لا يكتفي بالأرقام الأولية، بل يرسم التوزيعات، ويختبر الفرضيات، ويكشف الأنماط الخفية.

Each tool has its place in the bioinformatics workshop. Imagine the following scene:

اللغة المشتركة التي تربط هذه الأدوات جميعاً هي Python، لكن فهم دور كل منها يمنحك المرونة لاختيار الأداة المناسبة لكل مرحلة من مراحل التحليل.

· Bash: The site-preparation crew, managing files, executing tools, and connecting inputs to outputs in organized pipelines.

· Biopython (with Python): The smart engineer who understands the genetic blueprint, reads sequences, analyzes them, and performs transformations.

· R (with Bioconductor): The statistical inspector who does not settle for raw numbers, but plots distributions, tests hypotheses, and uncovers hidden patterns.

The common language binding these tools together is Python, but understanding the role of each gives you the flexibility to choose the right tool for each stage of analysis.

Bioethics and Algorithmic Biosecurity (Introduction).

9. الأخلاقيات الحيوية والسلامة الجينية البرمجية (مقدمة)

This preliminary chapter lays the first foundation for security awareness by introducing the principle of fail-closed. In sensitive systems, the default behavior should be denial or halting in the absence of explicit authorization or a trusted reference, rather than proceeding with implicit approval. We do not provide operational lists of pathogens here, but rather the defensive design philosophy that will grow in later parts.

هذا الفصل التمهيدي يضع أول حجر للوعي الأمني من خلال تقديم مبدأ fail-closed (الإغلاق عند الشك). في الأنظمة الحساسة، يجب أن يكون السلوك الافتراضي هو الرفض أو التوقف في حال غياب إذن صريح أو مرجع موثوق، وليس الاستمرار مع افتراض ضمني بالموافقة. لا نقدم هنا قوائم تشغيلية لمسببات الأمراض، بل نقدم فلسفة التصميم الدفاعي التي ستنمو في أبواب لاحقة.

```
# واجهة تعليمية رمزية فقط (لا توجد قواعد بيانات تشغيلية)
from biobrick_security import gene_drive_lock
# المحاكاة في وضع جاف (اختبار دون تنفيذ فعلي)
result = gene_drive_lock(dry_run=True, human_approval=False)
print(result["allowed"])  # سيكون False في غياب الموافقة الصريحة
```

Bridge to RNA-seq and Biostatistics.

Now that the environment is ready, and you understand object-oriented representation, it is time to remember that DNA is not just a static genome for editing; it is a living source of information that changes with conditions. The next part of the book will transition you from handling abstract sequences to analyzing gene expression data (RNA-seq), where you will read real files, normalize them, and discover genes that respond differently between samples. These bridging sections prevent bioinformatics from being reduced to a single tool, expanding the horizon toward statistical analysis and data visualization.

بعد أن أصبحت البيئة جاهزة، وفهمت التمثيل الكائني، حان الوقت للتذكير بأن الـ DNA ليس مجرد جينوم ثابت للتعديل، بل هو مصدر حي للمعلومات يتغير بتغير الظروف. الجزء التالي من الكتاب سينتقل بك من معالجة التسلسلات المجردة إلى تحليل بيانات التعبير الجيني (RNA-seq)، حيث ستقرأ الملفات الحقيقية، وتطبعها، وتكتشف الجينات التي تختلف استجابتها بين عينة وأخرى. هذه الجسور التمهيدية تمنع اختزال المعلوماتية الحيوية في أداة واحدة، وتوسع الأفق نحو التحليل الإحصائي والتصور البياني.

Foundation Chapter Outcome

مخرجات الفصل التمهيدي

بعد هذا الفصل، لن تحتاج فقط إلى تثبيت بايثون وتشغيله، بل ستفهم ماذا يحدث خلف الكواليس. ستعرف الفرق بين التسلسل كحروف، والتسلسل ككائن حيوي، والتسلسل كموقع على الخريطة الجينية. كما ستدرك أن تشغيل الكود ليس دليلاً على حقيقة بيولوجية، وأن إيقاف البرنامج طلباً للمراجعة قد يكون أكثر أماناً من الاستمرار في الظلام.

· تثبيت بيئة العمل والتحقق من صلاحيتها.

After this chapter, you will not only install and run Python—you will understand what happens behind the scenes. You will know the difference between a sequence as letters, a sequence as a biological object, and a sequence as a location on a genetic map. You will also realize that running code is not proof of a biological truth, and that stopping the program for review can be safer than continuing in the dark.

· فهم الأدوات الثلاث لتمثيل البيانات الجينية (Seq، SeqRecord، SeqFeature).

· Install and validate a working computational environment.

· التمييز بين الممارسة البرمجية أو المحاكاة، وبين الادعاء البيولوجي القائم على دليل.

· Understand the three core tools for representing genetic data (Seq, SeqRecord, and SeqFeature).

· معرفة أن التوقف عن التنفيذ عند فقدان الشرط، ورفع الأمر للمراجعة، هما سلوكان آمنان مقصودان، وليس مجرد أخطاء.

· Distinguish between a programming exercise or simulation, and a factual biological claim.

· Recognize that halting execution when conditions are not met, and escalating for review, are intentional safe behaviors, not mere errors.

Part 1 — Introduction to Bioinformatics

الباب 1 — المدخل إلى المعلوماتية الحيوية

الفكرة المركزية: تعريف المجال، السؤال العلمي، التمثيل الحسابي، الفرق بين الدليل الحسابي والإثبات البيولوجي.

Now that the programming environment is ready, and the preceding sections have established how to install Python, understand PATH, create a virtual environment, and test Biopython, it is time to ask a more fundamental question: What will we do with these tools? What is the field we are about to enter? This chapter provides the answer: bioinformatics. Bioinformatics is defined as an interdisciplinary scientific field that integrates biology with computer science and information technology. The National Center for Biotechnology Information (NCBI) defines bioinformatics as "the field of science in which biology, computer science, and information technology merge into a single discipline", and "the ultimate goal of the field is to enable the discovery of new biological insights as well as to create a global perspective from which unifying principles in biology can be discerned". The European Bioinformatics Institute (EBI) defines it as "the application of computer technology to the storage, management and analysis of biological data". Other sources indicate that it blends computer science and biostatistics with biological and biomedical sciences such as biochemistry, cell biology, genetics, genomics, and physiology.

Chapter 1 — What Is Bioinformatics

الفصل 1 — ما المعلوماتية الحيوية؟

This definition is not merely theoretical. When you search for a gene sequence in the GenBank database, the workflow is practically applying this definition: you use a computer to store data (the sequence in the database), then manipulate it (search for the sequence), then analyze it (compare it with other sequences). This is bioinformatics in its simplest form. The goal of bioinformatics is not merely to know a programming language, but to deeply understand how to choose the

appropriate data representation, the correct reference data, and the suitable statistical assumptions. These choices can change the scientific question itself and may lead to completely different conclusions. Bioinformatics begins with a clear biological question: What do we want to know about this gene? How does gene expression change between samples?

بعد أن أصبحت البيئة البرمجية جاهزة، وتعلمت كيف تثبت بايثون وتفهم PATH وتنشئ بيئة افتراضية وتختبر Biopython، حان الوقت لتسأل سؤالاً أكثر جوهرية: ما الذي سنفعله بهذه الأدوات؟ ما هو المجال الذي نستعد لخوض غماره؟ هذا الفصل يضع الإجابة: المعلوماتية الحيوية. تُعرَّف المعلوماتية الحيوية بأنها مجال علمي بيني (متعدد التخصصات) يدمج علم الأحياء مع علوم الحاسوب وتقانة المعلومات. يُعرِّف المركز الوطني لمعلومات التقانة الحيوية (NCBI) المعلوماتية الحيوية بأنها "مجال العلم الذي تندمج فيه الأحياء وعلوم الحاسوب وتقانة المعلومات في تخصص واحد"، والهدف الأسمى لهذا المجال هو "تمكين اكتشاف رؤى بيولوجية جديدة، بالإضافة إلى خلق منظور عالمي يمكن من خلاله تمييز المبادئ الموحدة في علم الأحياء". كما يعرّفها المعهد الأوروبي للمعلوماتية الحيوية (EBI) بأنها "تطبيق تقنية الحاسوب في تخزين البيانات البيولوجية وإدارتها وتحليلها". وتشير مصادر أخرى إلى أنها تمزج علوم الحاسوب والإحصاء الحيوي مع العلوم البيولوجية والطبية الحيوية مثل الكيمياء الحيوية وعلم الأحياء الخلوي وعلم الوراثة والجينوميات وعلم وظائف الأعضاء.

For example, the question "What is the similarity percentage between a specific gene in human and mouse?" directly translates into an alignment algorithm (such as BLAST) that compares the two sequences letter by letter. The question "How does gene expression change between a healthy and a diseased sample?" translates into a statistical analysis of reads in RNA-seq data. The question is the compass, and the algorithm is the tool that follows it. What is the evolutionary relationship between these organisms? This question is then translated into a language that computers understand, and data and algorithms are chosen based on the nature of the question, not the other way around.

هذا التعريف ليس نظرياً فقط. فعندما تبحث عن تسلسل جين في قاعدة بيانات GenBank، فإنك تطبق هذا التعريف عملياً: تستخدم الحاسوب لتخزين البيانات (التسلسل في قاعدة البيانات)، ثم تعالجها (تبحث عن التسلسل)، ثم تحللها (تقارنها بتسلسلات أخرى). هذه هي المعلوماتية الحيوية في أبسط صورها. ليس الهدف من المعلوماتية الحيوية مجرد معرفة لغة برمجة؛ بل الفهم العميق لكيفية اختيار التمثيل المناسب للبيانات، والبيانات المرجعية الصحيحة، والافتراضات الإحصائية الملائمة. فهذه الخيارات قد تغير السؤال العلمي نفسه، وقد تؤدي إلى استنتاجات مختلفة تماماً. تبدأ المعلوماتية الحيوية بسؤال بيولوجي واضح: ماذا نريد أن نعرف عن هذا الجين؟ كيف يتغير التعبير الجيني بين عينة وأخرى؟ ما العلاقة التطورية بين هذه الكائنات؟ مثلاً، السؤال "ما نسبة التشابه بين جين معين في الإنسان والفأر؟" يتحول مباشرة إلى خوارزمية محاذاة (مثل BLAST) تقارن التسلسلين حرفاً بحرف. والسؤال "كيف يتغير التعبير الجيني بين عينة سليمة وأخرى مريضة؟" يتحول إلى تحليل إحصائي للقراءات في بيانات RNA-seq.

السؤال هو البوصلة، والخوارزمية هي الأداة التي تتبعها. ثم يترجم هذا السؤال إلى لغة يفهمها الحاسوب، وتُختار البيانات والخوارزميات بناءً على طبيعة السؤال، وليس العكس.

Computational representation is the step that precedes computation itself. Before applying any algorithm, we must decide how to represent biological data in a digital format. Do we represent the sequence as plain text? Or as an intelligent biological object (like Seq in Biopython) that carries its direction and rules? Do we represent a gene as a location on a genetic map with start and end coordinates and strand orientation? The choice of representation determines what can be measured and what cannot. A good representation reveals hidden patterns, while a poor representation may hide them or create illusory patterns.

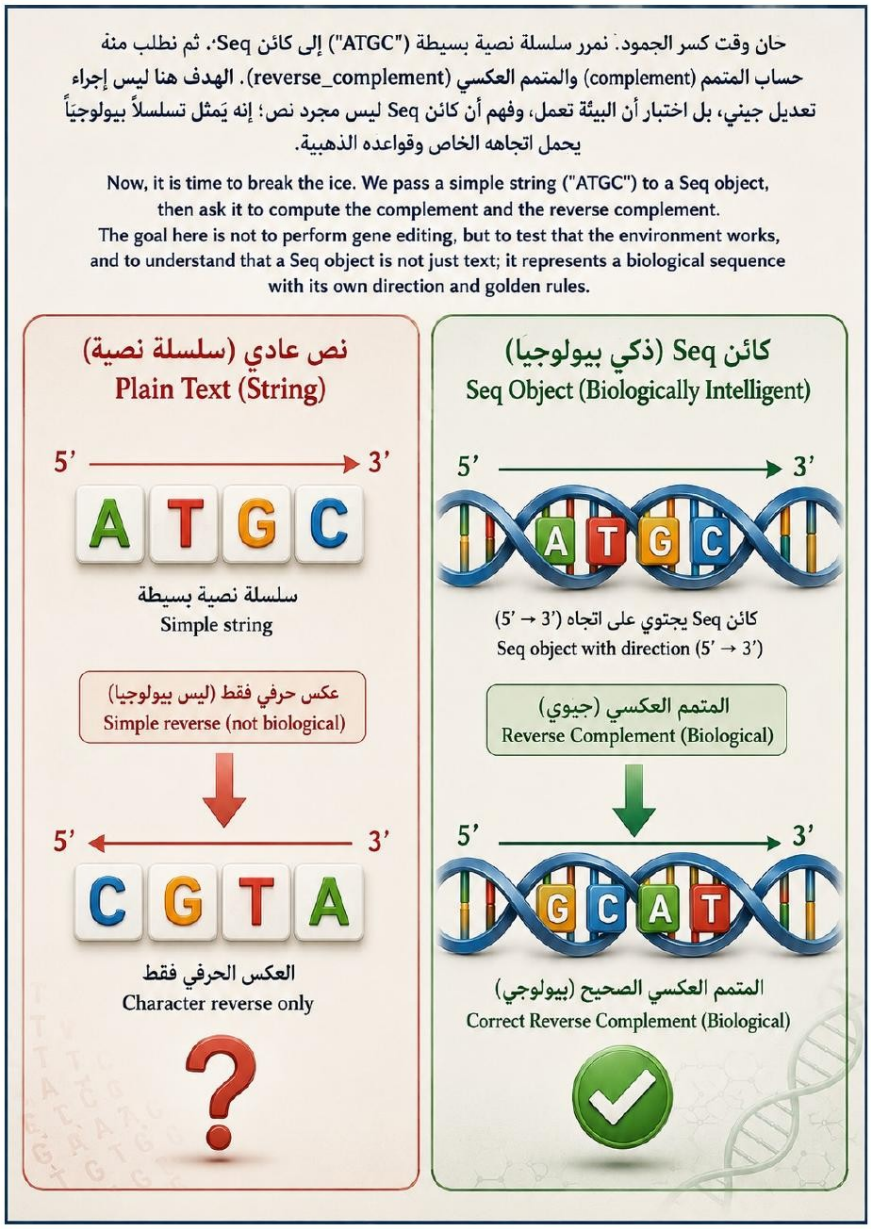
التمثيل الحسابي هو الخطوة التي تسبق الحساب نفسه. قبل أن نطبق أي خوارزمية، يجب أن نقرر كيف نمثل البيانات البيولوجية في صيغة رقمية مفهومة. هل نمثل التسلسل كنص عادي؟ أم ككائن حيوي ذكي (مثل Seq في Biopython) يحمل اتجاهه وقواعده؟ هل نمثل الجين كموقع على خريطة جينية بإحداثيات بداية ونهاية واتجاه؟ اختيار التمثيل يحدد ما يمكن قياسه وما لا يمكن قياسه. فالتمثيل الجيد يكشف الأنماط الخفية، والتمثيل السيء قد يخفيها أو يخلق أنماطاً وهمية.

Let us take a simple example: the sequence "ATGC". If you represent it as plain text, all the workflow can do is reverse it to become "CGTA". But if you represent it as a Seq object in Biopython, the workflow can compute its reverse complement to become "GCAT", which is the correct biological complement. The same letters, but two different representations give two completely different results. A good representation preserves biological meaning, while a poor representation loses it. This chapter distinguishes between computational description (such as measuring the degree of similarity between two sequences) and biological inference (such as proposing that a particular sequence performs a specific function or that two organisms are evolutionarily related). A pattern detected by software may generate a hypothesis, but its validation requires experimental, comparative, or independent evidence. High computational similarity between two sequences does not necessarily mean they perform the same function in the cell. Computational evidence is a powerful tool for generating hypotheses, A classic example in the scientific literature: Serine Proteases such as trypsin and chymotrypsin. Their sequences are highly similar, and they share the same catalytic mechanism, yet each has a different function in digestion. Computational similarity tells us they are related, but only laboratory experiments tell us the different function of each. but final biological proof requires laboratory experiments, functional studies, or

independent evidence from other sources. This distinction is the essence of the scientific method in bioinformatics.

لنأخذ مثالاً بسيطاً: تسلسل "ATGC". إذا مثلته كنص عادي، فكل ما يمكن فعله هو عكسه ليصبح "CGTA". لكن إذا مثلته ككائن Seq في Biopython، يمكن حساب مكمله العكسي ليصبح "GCAT" وهو المتمم البيولوجي الصحيح. نفس الحروف، ولكن تمثيلين مختلفين يعطيان نتيجتين مختلفتين تماماً. التمثيل الجيد يحافظ على المعنى البيولوجي، والتمثيل السيء يفقده. يميز هذا الفصل بين الوصف الحسابي (مثل قياس درجة التشابه بين تسلسلين) وبين الاستنتاج البيولوجي (مثل اقتراح أن تسلسلاً معيناً يؤدي وظيفة محددة أو أن كائنين مرتبطان تطورياً). فالنمط الذي يكتشفه البرنامج قد يولد فرضية، لكن إثباتها يتطلب أدلة تجريبية أو مقارنة أو مستقلة. قد يكون التشابه الحسابي مرتفعاً بين تسلسلين، لكن هذا لا يعني بالضرورة أنهما يؤديان الوظيفة نفسها في الخلية. من الأمثلة الكلاسيكية في الأدبيات العلمية: بروتينات السيرين بروتياز (Serine Proteases) مثل التربسين والكيموتريبسين. تتشابه تسلسلاتها بنسبة عالية، وتشترك في نفس الآلية التحفيزية، لكن لكل منهما وظيفة مختلفة في الهضم. التشابه الحسابي يخبرنا أنهما قريبان، لكن التجارب المعملية فقط هي التي تخبرنا بالوظيفة المختلفة لكل منهما. الدليل الحسابي أداة قوية لتوليد الفرضيات، لكن الإثبات البيولوجي النهائي يحتاج إلى تجارب مخبرية أو دراسات وظيفية أو أدلة مستقلة من مصادر أخرى. هذا التمييز هو جوهر المنهج العلمي في المعلوماتية الحيوية.

ومن الناحية المنهجية، يمكن النظر إلى المعلوماتية الحيوية كسلسلة تحويلات متصلة: سؤال علمي، ثم تعريف للمتغيرات، ثم اختيار للبيانات، ثم خوارزمية، ثم قياس لعدم اليقين، ثم تفسير. إذا تغيرت أي حلقة من هذه السلسلة، تغير معنى النتيجة. ولهذا تقتضي الممارسة العلمية في كل مرة طرح الأسئلة التالية: ما الذي قيس فعلاً؟ وما الذي لم يُقَس؟ وما الافتراض الذي سمح بالانتقال من البيانات إلى الاستنتاج؟ يمكن تلخيص هذه السلسلة في خمس درجات تصاعدية، كالسلم الذي لا يمكن تخطي درجة منه:



Infographic 3 / إنفوجرافيك 3

الدرجة الأولى: السؤال العلمي (ماذا نريد أن نعرف؟).

الدرجة الثانية: التمثيل (كيف نمثل البيانات؟).

Methodologically, bioinformatics can be viewed as a continuous chain of transformations: a scientific question, followed by variable definition, data selection, algorithm application, uncertainty measurement, and finally interpretation. If any link in this chain changes, the meaning of the result changes. This is why each analytical step should be accompanied by the following questions: What was actually measured? What was not measured? And what assumption allowed the transition from data to conclusion? This chain can be summarized in five ascending steps, like a ladder where no step can be skipped:

الدرجة الثالثة: الحساب (أي خوارزمية نطبق؟).

Step one: The scientific question (What do we want to know?).

الدرجة الرابعة: التفسير (ماذا تعني النتائج؟).

Step two: The representation (How do we represent the data?).

الدرجة الخامسة: التحقق (هل هذا الاستنتاج مدعوم بأدلة مستقلة؟).

Step three: The computation (Which algorithm do we apply?).

Step four: The interpretation (What do the results mean?).

Step five: The validation (Is this conclusion supported by independent evidence?).

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "What is Bioinformatics?" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "ما المعلوماتية الحيوية؟" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "computational similarity proves biological function". This oversteps the limits of computational evidence. The correct statement is: "computational similarity prioritizes genes or proteins for further study, but it does not prove their function." Bioinformatics generates testable hypotheses, not final biological truths. A simple test: after any computational analysis, Practical check: "Does this result tell me about a real function in the cell, or is it merely a numerical indicator that needs laboratory confirmation?".

مثال على التجاوز: القول إن "التشابه الحسابي يثبت الوظيفة البيولوجية". فهذا قول يتجاوز حدود الدليل الحسابي.

المطلوب هو القول إن "التشابه الحسابي يرشح جينات أو بروتينات لدراسة إضافية، لكنه لا يثبت وظيفتها".

فالمعلوماتية الحيوية تولد فرضيات قابلة للاختبار، وليست حقائق بيولوجية نهائية. اختبار بسيط: فحص تطبيقي بعد

أي تحليل حسابي: "هل هذه النتيجة تخبرني عن وظيفة حقيقية في الخلية، أم أنها مجرد مؤشر رقمي يحتاج إلى تأكيد معملي؟"ـ

2. Computational representation versus biological claim — Distinguish computational representation from biological claims

2. التمثيل الحسابي مقابل الادعاء البيولوجي — تمييز التمثيل الحسابي عن الادعاء البيولوجي

Example: calculating the percentage similarity between two sequences (e.g., 85%) is purely computational representation. Claiming that "these two sequences perform the same function because they are 85% similar" is an unjustified biological claim. Computational similarity measures letter identity only; it does not measure cellular context or molecular mechanism. Practical test: before converting any computational result into a biological claim, look for independent evidence (such as laboratory experiments, functional studies, or a reliable scientific reference) that supports this claim.

مثال: حساب النسبة المئوية للتشابه بين تسلسلين (مثل 85%) هو تمثيل حسابي بحت. أما القول بأن "هذين التسلسلين يؤديان الوظيفة نفسها لأن تشابههما 85%" فهو ادعاء بيولوجي غير مبرر. التشابه الحسابي يقيس تطابق الحروف فقط، ولا يقيس السياق الخلوي أو الآلية الجزيئية. اختبار عملي: قبل أن تتحول أي نتيجة حسابية إلى ادعاء بيولوجي، ابحث عن دليل مستقل (مثل تجربة معملية أو دراسة وظيفية أو مرجع علمي موثوق) يدعم هذا الادعاءـ

3. المدخلات والافتراضات والمخرجات والقيود — تحديد المدخلات والافتراضات والمخرجات والقيود

مثال تطبيقي على أداة BLAST:

· المدخلات: تسلسل DNA خام، وقاعدة بيانات مرجعية (مثل GenBank)ـ

· الافتراضات: جودة البيانات، خلوه من أخطاء التسلسل، واختيار مصفوفة الاستبدال المناسبة (مثل BLOSUM62 للبروتينات)ـ

· المخرجات: قائمة بالمطابقات المحتملة مع درجات الثقة (E-value)ـ

3. Inputs, assumptions, outputs, and limitations — Identify inputs, assumptions, outputs, and limitations

· القيود: قد تفوت الخوارزمية تسلسلات بعيدة التشابه، وقد تعطي نتائج إيجابية خاطئة إذا كانت قاعدة البيانات غير مكتملة أو كان التسلسل المدخل منخفض الجودةـ

Applied example using BLAST:

اختبار: قبل تشغيل أي تحليل، اكتب قائمة واضحة بهذه العناصر الأربعة. إذا لم تستطع تحديد أحدها، فأنت لست جاهزاً للتشغيل.

· Inputs: raw DNA sequence and a reference database (e.g., GenBank).

· Assumptions: data quality, freedom from sequencing errors, and choosing the appropriate substitution matrix (e.g., BLOSUM62 for proteins).

· Outputs: list of potential matches with confidence scores (E-value).

· Limitations: the algorithm may miss distantly related sequences, and may produce false positives if the database is incomplete or the input sequence is low quality.

Test: before running any analysis, write a clear list of these four elements. If you cannot identify one of them, the workflow is not ready to run.

4. المنشأ والإصدار والمعلمات والتحويلات — توثيق المصدر والإصدار والمعلمات والتحويلات اللازمة لإعادة الإنتاج

مثال: عند تحليل تسلسل، يجب تسجيل:

· المصدر: رقم الانضمام في قاعدة البيانات (مثل NM_000660 لجين TGFB1 البشري).

· الإصدار: إصدار بايثون (مثل 3.11)، وإصدار Biopython (مثل 1.81).

· المعلمات: قيمة العتبة (مثل E-value < 0.001)، ومصفوفة الاستبدال المستخدمة.

4. Provenance, versions, parameters, and transformations — Record provenance, versions, parameters, and transformations required for reproducibility

· التحويلات: خطوات المعالجة التي طُبقت على البيانات الخام (مثل إزالة البادئات، أو قص التسلسل، أو تغيير الحالة إلى أحرف كبيرة).

Example: when analyzing a sequence, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا الملف لزميل بعد عام، هل يستطيع إعادة بناء النتيجة خطوة بخطوة من دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: accession number in the database (e.g., NM_000660 for human TGFB1 gene).

· Version: Python version (e.g., 3.11), Biopython version (e.g., 1.81).

· Parameters: threshold value (e.g., E-value < 0.001), substitution matrix used.

· Transformations: processing steps applied to the raw data (e.g., removing prefixes, trimming the sequence, or converting to uppercase).

Test: Practical check: "If I gave this file to a colleague a year later, could they rebuild the result step by step without my help?" If the answer is no, the documentation is insufficient.

5. الاختبار الاصطناعي قبل الاعتماد على البيانات الحقيقية — استخدام اختبارات اصطناعية ومحايدة قبل الاعتماد على بيانات حقيقية

5. Synthetic testing before reliance on real data — Use synthetic or neutral tests before relying on real data

مثال: قبل تشغيل أي كود على بيانات جينومية حقيقية (مثل تسلسل مريض)، اختبر الكود على تسلسل اصطناعي صغير معروف النتيجة مسبقاً. مثلاً، استخدم التسلسل "ATGC" وتأكد أن دالة المتمم العكسي تعطي "GCAT". إذا نجح الاختبار على البيانات الاصطناعية، فأنت واثق من أن الكود يعمل كما هو متوقع. أما إذا فشل، فستعرف أن المشكلة في الكود وليس في البيانات الحقيقية. هذه الممارسة توفر ساعات من البحث عن أخطاء وهمية في بيانات حقيقية معقدة.

Example: before running any code on real genomic data (e.g., a patient's sequence), test the code on a small synthetic sequence with a known outcome. For instance, use the sequence "ATGC" and verify that the reverse complement function returns "GCAT". If the test succeeds on synthetic data, the workflow is confident the code works as expected. If it fails, you know the problem lies in the code, not in the real data. This practice saves hours of searching for phantom errors in complex real data.

هذه الأسئلة الخمسة مترابطة؛ فهي تحول الفكرة المركزية إلى طريقة عملية للفحص والتوثيق والاختبار. فهي تعيد صياغة الفكرة المركزية للفصل بأسلوب عملي: تعريف المجال (المخرج الأول)، والتمييز بين التمثيل والادعاء (المخرج الثاني)، وتحديد المدخلات والقيود (المخرج الثالث)، وتوثيق الخطوات (المخرج الرابع)، والاختبار قبل التنفيذ (المخرج الخامس). كل مخرج يبني على الذي قبله، وكلها تصب في هدف واحد: فهم ما هي المعلوماتية الحيوية، وما هي حدودها، وكيف نمارسها بأمانة علمية. وقبل أن نختتم هذا الفصل، هناك قاعدة منهجية مهمة: لا تثق بأي كود قبل أن تختبره على بيانات اصطناعية صغيرة معروفة النتيجة. جرب دائماً على تسلسل بسيط مثل "ATGC" وتأكد أن ناتج المتمم العكسي هو "GCAT" قبل أن تنتقل إلى تسلسل مريض حقيقي. هذه العادة البسيطة توفر ساعات من البحث عن أخطاء في بيانات معقدة.

These five questions are connected; together they turn the central idea into a practical method for checking, documenting, and testing the analysis. They reformulate the chapter's central idea in a practical manner: defining the field (outcome one), distinguishing representation from claims (outcome two),

identifying inputs and limitations (outcome three), documenting steps (outcome four), and testing before execution (outcome five). Each outcome builds on the previous one, and all serve a single goal: understanding what bioinformatics is, what its limits are, and how to practice it with scientific integrity. And before we conclude this chapter, there is an important methodological rule: do not trust any code before testing it on small synthetic data with a known outcome. Always test on a simple sequence like "ATGC" and verify that the reverse complement output is "GCAT" before moving to a real patient sequence. This simple habit saves hours of searching for errors in complex data.

Chapter 2 — From DNA to a Data File

الفصل 2 — من DNA إلى ملف بيانات

الفكرة المركزية: التسلسل والهوية والبيانات الوصفية والمنشأ والصيغ.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

تحويل الجزيء إلى ملف يخلق طبقة تمثيل جديدة. الحروف A وT وC وG تحفظ ترتيباً، لكنها لا تحفظ وحدها السياق الخلوي أو البنية ثلاثية الأبعاد أو حالة العينة. لذلك تصبح الهوية والمنشأ والإصدار والبيانات الوصفية جزءاً من المعنى العلمي للملف، لا إضافات شكلية.

Turning a molecule into a file creates a new representation layer. The letters A, T, C, and G preserve order, but by themselves they do not preserve cellular context, three-dimensional structure, or sample state. Identity, provenance, version, and metadata therefore become part of the scientific meaning of the file rather than decorative additions.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

Now that you know what bioinformatics is, and you understand it is the art of translating biological questions into a language computers understand, it is time for the first practical question: Where does the journey actually begin? How does living DNA in the cell, which is a complex molecule that interacts with proteins and is influenced by the cellular environment, transform into a simple text file you store on the project’s computer's hard driv?e.

بعد أن عرفت ما هي المعلوماتية الحيوية، وفهمت أنها فن ترجمة الأسئلة البيولوجية إلى لغة يفهمها الحاسوب، تبدأ هنا مرحلة السؤال العملي الأول: من أين تبدأ الرحلة فعلياً؟ كيف يتحول الـ DNA الحي في الخلية، والذي هو جزيء معقد يتفاعل مع البروتينات ويتأثر بالبيئة الخلوية، إلى ملف نصي بسيط تخزنه على قرص حاسوبك؟

This question is the essence of Chapter 2. The short answer is: DNA in a file is not DNA in the cell. DNA in a file is a photograph, not the molecule itself. It is merely a string of four letters (A, T, C, G), having lost everything that surrounds it in the cell: associated proteins, chemical modifications, three-dimensional structure, and the spatial and temporal context of gene expression. The transition from molecule to file is not just writing letters; it is a loss of biological context and a gain of a form that computers can process. This transformation is the first and deepest challenge in bioinformatics: how do you preserve scientific meaning while stripping data of everything that makes it aliv?e.

هذا السؤال هو جوهر الفصل الثاني. الإجابة المختصرة هي: الـ DNA في الملف ليس الـ DNA في الخلية. الـ DNA في الملف هو صورة فوتوغرافية، وليس الجزيء نفسه. هو سلسلة من الحروف الأربعة (A، T، C، G) فقط، وقد فقد كل ما يحيط به في الخلية: البروتينات المرتبطة به، التعديلات الكيميائية، البنية ثلاثية الأبعاد، والسياق الزماني والمكاني للتعبير الجيني. الانتقال من الجزيء إلى الملف ليس مجرد كتابة حروف؛ إنه فقدان للسياق الحيوي واكتساب لشكل يمكن للحاسوب معالجته. هذا التحول هو أول وأعمق تحدٍ في المعلوماتية الحيوية: كيف تحافظ على المعنى العلمي وأنت تجرد البيانات من كل ما يجعلها حية؟

But a good scientific record does not stop at naked letters. It links the sequence to a unique identity, a specific source, a precise description, a known version, and a documented date. For illustration, consider a case in which find a piece of paper with "ATGC" written on it. What does it mean? Nothing. But if you find a paper that says "this sequence is part of the human insulin gene, extracted from an American tissue bank in 2023," suddenly the paper has value. This is the difference between raw data and scientific knowledge. Two sequences that are literally identical may be scientifically different if one has documented provenance and the other does not. Losing provenance means losing trust, and losing trust means the data becomes unusable for any serious scientific conclusion.

لكن السجل العلمي الجيد لا يكتفي بالحروف العارية. فهو يربط التسلسل بهوية فريدة، ومصدر محدد، ووصف دقيق، وإصدار معروف، وتاريخ موثق. للتوضيح، يمكن تصور حالة يكون فيها تجد ورقة مكتوب عليها "ATGC" فقط. ماذا تعني؟ لا شيء. لكن إذا وجدت ورقة مكتوب عليها "هذا التسلسل هو جزء من جين الأنسولين البشري، استُخرج من بنك أمريكي للأنسجة في عام 2023"، فجأة أصبحت الورقة ذات قيمة. هذا هو الفرق بين البيانات الخام والمعرفة

العلمية. قد يكون تسلسلان متطابقان حرفياً مختلفين علمياً إذا كان أحدهما موثق المصدر والآخر مجهول المنشأ. فقدان المنشأ يعني فقدان الثقة، وفقدان الثقة يعني أن البيانات تصبح غير قابلة للاستخدام في أي استنتاج علمي جاد.

Metadata is not marginal, nor is it an academic luxury. Sample type, organism, tissue, sequence source, the method by which it was obtained (such as the sequencing technology used, Sanger or Illumina), and the reference version all determine the validity of any comparison. When this context is absent, the analysis may be computationally correct but scientifically uninterpretable. A simple example: if you compare a sequence from a patient's tissue with a sequence from healthy tissue, but you did not document that both sequences came from the same type of cell, the comparison may be meaningless. The difference you see may be a difference between cell types, not between health and disease.

البيانات الوصفية ليست هامشاً، وليست ترفاً أكاديمياً. نوع العينة، الكائن الحي، النسيج، مصدر التسلسل، طريقة الحصول عليه (مثل تقنية التسلسل المستخدمة، Sanger أم Illumina)، والإصدار المرجعي كلها عناصر تحدد صلاحية المقارنة. عندما يغيب هذا السياق، قد يكون التحليل صحيحاً حسابياً لكنه غير قابل للتفسير علمياً. مثال بسيط: لو قارنت تسلسلاً من أنسجة مريض بتسلسل من أنسجة سليمة، لكنك لم توثق أن التسلسلين أتيا من نفس النوع من الخلايا، فقد تكون المقارنة غير ذات معنى. فالاختلاف الذي تراه قد يكون اختلافاً بين أنواع الخلايا، وليس بين الصحة والمرض.

The first step in any bioinformatics project is to convert the raw sequence into an organized file that carries this information. The simplest and most widely used format in the world is FASTA. This format is like a simple envelope: the first line is the header, starting with a ">" and carrying the identity and description, and the following lines carry the sequence itself. Why is this format so simple? Because it was designed to be readable by both humans and machines, and because it requires no special software to open, only any text editor. This simple format teaches a profound lesson: a sequence has no scientific value without its identity. Identity is what makes it searchable, citable, and comparable.

أول خطوة في أي مشروع معلوماتي حيوي هي تحويل التسلسل الخام إلى ملف منظم يحمل هذه المعلومات. الصيغة الأكثر بساطة وانتشاراً في العالم هي FASTA. هذه الصيغة تشبه ظرفاً بسيطاً: السطر الأول هو العنوان، ويبدأ بعلامة ">" ويحمل الهوية والوصف، والسطور التالية تحمل التسلسل نفسه. لماذا هذه الصيغة بهذه البساطة؟ لأنها صُممت لتكون مقروءة من البشر والآلات معاً، ولأنها لا تحتاج إلى برامج خاصة لفتحها، فقط أي محرر نصوص. هذا التنسيق البسيط يوضح درساً عميقاً: التسلسل لا قيمة علمية له دون هويته. فالهوية هي ما يجعله قابلاً للبحث، والاستشهاد، والمقارنة.

Let us take a concrete example: for illustration, consider a case in which have the sequence "ATGC" without any additional information. Merely letters, meaningless, and you cannot do anything with them in a scientific context. But if the workflow has the following file.

لنأخذ مثالاً ملموساً: تخيل أن يوجد في المثال تسلسل "ATGC" دون أي معلومات إضافية. مجرد حروف، ليس لها معنى، ولا يمكن أن تفعل بها شيئاً في سياق علمي. لكن إذا كان يوجد في المثال الملف التالي:

seq_001 | Homo sapiens | insulin.

```
ATGC
```

then the workflow has a data file that carries meaning: this sequence came from a human (Homo sapiens), from the insulin gene, and has a unique identity (seq_001). The difference between the two files is the difference between random letters and scientific knowledge. In the second case, the workflow can now search for this sequence in databases, compare it to other sequences, cite it in a research paper, and reanalyze it years later.

فقد أصبح يوجد في المثال ملف بيانات يحمل معنى: هذا التسلسل جاء من إنسان (Homo sapiens)، ومن جين الأنسولين، وله هوية فريدة (seq_001). الفرق بين الملفين هو الفرق بين فوضى الحروف والمعرفة العلمية. في الحالة الثانية، يمكن الآن البحث عن هذا التسلسل في قواعد البيانات، ومقارنته بتسلسلات أخرى، والاستشهاد به في ورقة بحثية، وإعادة تحليله بعد سنوات.

This transition from molecule to file, from letters to record, is what gives data its scientific value. The file is not just a container; it is a document carrying the sequence's birth certificate: Where did it come from? When? What is it? Only with this certificate does the sequence become a scientific tool that can be referenced, compared, and reanalyzed. Without this certificate, the sequence becomes mere digital noise.

هذا الانتقال من الجزيء إلى الملف، ومن الحروف إلى السجل، هو ما يمنح البيانات قيمتها العلمية. الملف ليس مجرد وعاء؛ بل وثيقة تحمل شهادة ميلاد التسلسل: من أين جاء؟ متى؟ ما هو؟ وبهذه الشهادة فقط يصبح التسلسل أداة علمية يمكن الرجوع إليها ومقارنتها وإعادة تحليلها. بدون هذه الشهادة، يصبح التسلسل مجرد ضوضاء رقمية.

Let us apply this practically. Before you start analyzing real FASTA files from databases like GenBank, create a small synthetic file by hand. Write it in a text editor:

لنطبق هذا عملياً. قبل أن تبدأ في تحليل ملفات FASTA حقيقية من قواعد البيانات مثل GenBank، اصنع ملفاً اصطناعياً صغيراً بيدك. اكتبه في محرر نصوص:

synthetic_001 | test

```
ATGC
```

Then try to read it using Biopython. Write simple code that reads this file, extracts the identity and sequence, and prints them to the screen. If the code succeeds with this synthetic file, the workflow is confident that the project's tools work as expected. If it fails, you know the problem lies in the code or in how you wrote the file, not in the real data. This simple habit saves hours of searching for errors in complex data. It tests not only the project's code but also the project's understanding of how data is represented. It is a practical embodiment of the golden rule: test on the simple before you face the complex.

ثم حاول قراءته باستخدام Biopython. اكتب كوداً بسيطاً يقرأ هذا الملف، ويستخرج الهوية والتسلسل، ويطبعها على الشاشة. إذا نجح الكود مع هذا الملف الاصطناعي، فأنت واثق من أن أدواتك تعمل كما هو متوقع. أما إذا فشل، فستعرف أن المشكلة في الكود أو في طريقة كتابة الملف، وليس في البيانات الحقيقية. هذه العادة البسيطة توفر ساعات من البحث عن أخطاء في بيانات معقدة. إنها لا تختبر الكود فحسب، بل تختبر فهمك لكيفية تمثيل البيانات. وهي تجسيد عملي للقاعدة الذهبية: اختبر على البسيط قبل أن تواجه المعقد.

When working with any data file, you must identify four fundamental elements as a framework for critical thinking. Inputs: What data does the program need to work? In the case of FASTA, it is the raw sequence, identity, and description. Assumptions: What does the program or file assume about the data? It assumes that the sequence is written with the correct letters A, T, C, G only, and that the identity is unique and not duplicated in the same file. Outputs: What does the program produce after processing? A simple text file readable by any editor. Limitations: What cannot this file or program do? A FASTA file does not carry quality information, nor genetic coordinates, nor other features such as coding regions or translation start sites. Identifying these elements before starting analysis protects you from the trap of misleading results.

عند التعامل مع أي ملف بيانات، يجب أن تحدد أربعة عناصر أساسية كإطار للتفكير النقدي. المدخلات: ما هي البيانات التي يحتاجها البرنامج ليعمل؟ في حالة FASTA، هي التسلسل الخام والهوية والوصف. الافتراضات: ما الذي يفترضه البرنامج أو الملف عن البيانات؟ يفترض أن التسلسل مكتوب بالحروف الصحيحة A، T، C، G فقط، وأن الهوية فريدة وغير مكررة في نفس الملف. المخرجات: ماذا ينتج البرنامج بعد المعالجة؟ ملف نصي بسيط يمكن قراءته بأي محرر. القيود: ما الذي لا يستطيع هذا الملف أو البرنامج فعله؟ ملف FASTA لا يحمل معلومات عن جودة القراءات، ولا عن الإحداثيات الجينية، ولا عن الميزات الأخرى مثل المناطق المشفرة أو مواقع بدء الترجمة. تحديد هذه العناصر قبل بدء التحليل يحميك من الوقوع في فخ النتائج المضللة.

When creating the project’s own file, be meticulous in documenting everything. Record the provenance: from which organism, which tissue, and by what method was the sequence extracted. Record the version: the file creation date, or the reference version number you relied on. Record the parameters: the sequencing technology used, and any other technical parameters. Record the transformations: any modifications applied to the raw sequence, such as removing prefixes, trimming unwanted parts, or converting letters to uppercase. This documentation is what makes the result reproducible. Imagine reanalyzing the project’s data a year later, or a colleague trying to replicate the project’s analysis. Without this record, the project’s steps will seem like magic, not science.

There is another golden rule no less important: do not trust any file before you know its source. The same sequence may be stored in multiple databases, each with a different version, date, and context. Using the wrong or outdated sequence may completely change the project’s analysis. For example, a particular gene may have been corrected in the database after the discovery of an error in the original sequence; if you use the old version, you will build the project’s analysis on a mistake. Therefore, always keep a record of the source, version, and transformations you performed. This habit is not mere organization; it is the essence of scientific integrity in the digital age.

عند إنشاء ملفك الخاص، احرص على توثيق كل شيء. سجل المصدر: من أي كائن حي، من أي نسيج، وبأي طريقة استُخرج التسلسل. سجل الإصدار: تاريخ إنشاء الملف، أو رقم النسخة المرجعية التي اعتمدت عليها. سجل المعلمات: تقنية التسلسل المستخدمة، وأي معاملات تقنية أخرى. سجل التحويلات: أي تعديلات طبقت على التسلسل الخام، مثل إزالة البادئات، أو قص أجزاء غير مرغوب فيها، أو تحويل الحروف إلى أحرف كبيرة. هذا التوثيق هو ما يجعل النتيجة قابلة لإعادة الإنتاج. للتوضيح، يمكن تصور حالة يكون فيها تعيد تحليل بياناتك بعد عام، أو أن زميلاً لك يحاول تكرار تحليلك. بدون هذا السجل، ستبدو خطواتك وكأنها سحر، وليس علماً.

Fifth Outcome: Use synthetic or neutral tests before relying on real data

مدخلات التسلسل
Sequence Input
A T C G
X N
حرف غير متوقع
Unexpected Character
خطأ في القيمة
ValueError
A T C G
تسلسل صالح
Valid Sequence
صحيح
Valid / Correct
(البرمجة الدفاعية).
البرمجة الدفاعية تبدأ بالتحقق من صحة المدخلات.
قبل إجراء أي عملية على تسلسل، يجب التأكد من أنه
يحتوي على الحروف المسموحة فقط (A، T، C، G).
استخدام دالة مثل set(sequence).issubset('ATCG')
يحمي الكود من الانهيار بسبب حرف غير متوقع،
ويحوّل الخطأ المحتمل إلى رسالة واضحة.
هذا هو التجسيد البرمجي لمبدأ "فشل آمن"
(Fail-securely) قبل أن نصل إلى البوابات
الأمنية المتقدمة.
(Defensive Programming).
Defensive programming begins with input validation. Before performing any operation on a sequence, ensure it contains only allowed characters (A, T, C, G). Using a function like `set(sequence).issubset('ATCG')` protects the code from crashing due to an unexpected character, turning a potential error into a clear message. This is the coding embodiment of the "fail-securely" principle before we even reach the advanced security gates.
المدخلات
Input
A T C G
حروف مسموحة
Allowed Characters
التحقق
Validation
يتم السماح فقط
بالحروف المسموحة
Only allowed characters
are permitted
مرفوض (خطأ)
Rejected (Error)
X N
حرف غير متوقع
Unexpected Character
خطأ في القيمة
ValueError
مقبول (صحيح)
Accepted (Valid)
A T C G
تسلسل صالح
Valid Sequence

Infographic 2 / إنفوجرافيك 2

In summary: from DNA to a data file is not merely a technical conversion; it is a methodological decision carrying deep scientific assumptions. Every step in this conversion carries an assumption, and every assumption may change the result. Understanding this process means the workflow is no longer just a user of

bioinformatics tools; the workflow is now able to evaluate them, critique them, and build the project's analyses on them with confidence. You now know that the file is not the molecule, that the representation is not the truth, and that documentation is the backbone of all reproducible science.

هناك قاعدة ذهبية أخرى لا تقل أهمية: لا تثق بأي ملف قبل أن تعرف مصدره. قد يكون التسلسل نفسه مخزناً في قواعد بيانات متعددة، لكن لكل منها إصدار وتاريخ وسياق مختلف. استخدام التسلسل الخاطئ أو القديم قد يغير نتيجة تحليلك بالكامل. فمثلاً، قد يكون جين معين قد أعيد تصحيحه في قاعدة البيانات بعد اكتشاف خطأ في التسلسل الأصلي، فإذا استخدمت النسخة القديمة، فستبني تحليلك على خطأ. لذلك، احتفظ دائماً بسجل للمصدر والإصدار والتحويلات التي أجريتها. هذه العادة ليست مجرد تنظيم، بل هي جوهر النزاهة العلمية في العصر الرقمي.

الخلاصة: من DNA إلى ملف بيانات ليس مجرد تحويل تقني، بل هو قرار منهجي يحمل في طياته افتراضات علمية عميقة. كل خطوة في هذا التحويل تحمل افتراضاً، وكل افتراض قد يغير النتيجة. فهم هذه العملية يعني أنك لم تعد مجرد مستخدم لأدوات المعلوماتية الحيوية، بل أصبحت قادراً على تقييمها ونقدها وبناء تحليلاتك عليها بثقة. أنت الآن تعرف أن الملف ليس الجزيء، وأن التمثيل ليس الحقيقة، وأن التوثيق هو العمود الفقري لكل علم قابل للتكرار.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "From DNA to a Data File" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "من DNA إلى ملف بيانات" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "the file contains all the information about the molecule." This is incorrect because the file contains only the sequence of letters and some metadata. The correct statement: "the file contains a representation of the sequence with some basic metadata, but it does not contain the cellular context, three-dimensional structure, or biological dynamics of the molecule." Accurate understanding of this distinction is the foundation of every sound bioinformatics analysis.

مثال على التجاوز: القول إن "الملف يحوي كل المعلومات عن الجزيء". هذا قول خاطئ لأن الملف يحوي فقط تسلسل الحروف وبعض البيانات الوصفية. الصحيح: "الملف يحوي تمثيلاً للتسلسل مع بعض البيانات الوصفية الأساسية، لكنه لا يحوي السياق الخلوي، أو البنية ثلاثية الأبعاد، أو الديناميكية الحيوية للجزيء." الفهم الدقيق لهذا التمييز هو أساس كل تحليل معلوماتي حيوي سليم.

Example: The presence of a sequence in a FASTA file does not mean that this sequence performs a specific function in the cell. The file only tells us about the order of letters, not about the biological role of the molecule. Claiming that "this

sequence is the insulin gene" requires additional experimental evidence, such as gene expression experiments or functional studies, not just the presence of the sequence in a file. Distinguishing between what the file tells us and what we infer about the living organism is the essence of critical scientific methodology.

مثال: وجود تسلسل في ملف FASTA لا يعني أن هذا التسلسل يؤدي وظيفة محددة في الخلية. الملف يخبرنا فقط عن ترتيب الحروف، وليس عن الدور الحيوي للجزيء. الادعاء بأن "هذا التسلسل هو جين الأنسولين" يحتاج إلى أدلة تجريبية إضافية، مثل تجارب التعبير الجيني أو دراسات الوظيفة، وليس فقط وجود التسلسل في ملف. الفصل بين ما يخبرنا به الملف وما نستنتجه عن الكائن الحي هو جوهر المنهج العلمي النقدي.

مثال تطبيقي على ملف FASTA:

· المدخلات: تسلسل خام، هوية فريدة (مثل اسم الجين أو رقم الانضمام)، ووصف مختصر.

· الافتراضات: أن التسلسل مكتوب بالحروف الصحيحة (A، T، C، G) فقط، وأن الهوية فريدة ضمن الملف، وأن السطر الأول يبدأ بعلامة ">"، وأن السطور التالية تحمل التسلسل.

· المخرجات: ملف نصي بسيط يمكن قراءته وتحريره بأي محرر نصوص، وقابل للاستخدام في معظم أدوات المعلوماتية الحيوية.

Applied example using a FASTA file:

· القيود: لا يحمل معلومات عن جودة القواعد، ولا عن الإحداثيات الجينية، ولا عن الميزات كالمناطق المشفرة، ولا عن السياق الخلوي. هذه القيود تجعل FASTA مناسباً للمقارنات البسيطة ولكنه غير كافٍ للتحليلات المتقدمة التي تحتاج إلى بيانات الجودة أو الإحداثيات.

· Inputs: raw sequence, unique identity (e.g., gene name or accession number), and a brief description.

· Assumptions: that the sequence is written with the correct letters (A, T, C, G) only, that the identity is unique within the file, that the first line starts with a ">", and that the following lines carry the sequence.

· Outputs: a simple text file that can be read and edited with any text editor, and is usable in most bioinformatics tools.

· Limitations: it does not carry base quality information, nor genetic coordinates, nor features such as coding regions, nor cellular context. These limitations make FASTA suitable for simple comparisons but insufficient for advanced analyses requiring quality or coordinate data.

عند إنشاء ملف FASTA، يجب تسجيل ما يلي:

· المصدر: الكائن الحي، النسيج، طريقة الاستخراج، والمختبر أو المركز الذي أنتج التسلسل.

· الإصدار: تاريخ الإنشاء، أو رقم النسخة المرجعية من قاعدة البيانات (مثل رقم الإصدار في GenBank).

· المعلمات: تقنية التسلسل المستخدمة (مثل Sanger، Illumina، PacBio)، ونوعية القراءات إذا كانت متوفرة.

· التحويلات: أي تعديلات طُبقت على التسلسل الخام، مثل إزالة البادئات (primers)، أو قص التسلسل إلى جزء معين، أو تصحيح الأخطاء، أو تغيير الحالة إلى أحرف كبيرة.

When creating a FASTA file, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا الملف لزميل بعد عام، مع نسخة من الكود الذي استخدمته، هل يستطيع أن يفهم كل خطوة قمت بها ويعيد بناء النتيجة من الصفر؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ. التوثيق الجيد ليس مجرد كرم، بل هو واجب علمي.

· Provenance: organism, tissue, extraction method, and the laboratory or center that produced the sequence.

· Version: creation date, or reference version number from the database (e.g., version number in GenBank).

· Parameters: sequencing tech

Chapter 3 — Data Quality and Reproducibility

الفصل 3 — جودة البيانات وإمكانية إعادة الإنتاج

Deeper Conceptual Explanation

الفكرة المركزية: المنشأ، إدارة الإصدارات، سلامة البيانات، سجل التدقيق، وإعادة بناء القرار.

شرح أعمق للمفهوم

Data quality is not a cosmetic stage that precedes analysis and is then forgotten. A small input error can propagate through every stage and appear at the end as a convincing-looking conclusion. Quality control is therefore tied to provenance, transformations, versions, and parameters; reproducibility means more than rerunning code—it means rebuilding the conditions that produced the result.

جودة البيانات ليست مرحلة تجميلية تسبق التحليل ثم تُنسى. خطأ صغير في الإدخال قد ينتقل عبر جميع المراحل ويظهر في النهاية كاستنتاج يبدو مقنعاً. لهذا ترتبط مراقبة الجودة بإمكانية تتبع مصدر البيانات والتحويلات والإصدارات والمعلمات؛ فإعادة الإنتاج لا تعني فقط إعادة تشغيل الكود، بل إعادة بناء الشروط التي أنتجت النتيجة.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Data quality is not a vague concept or an academic luxury. It is simply "the degree to which data is fit for its intended purpose." Data may be perfectly accurate for one purpose, yet completely unsuitable for another. For illustration, consider a case in which have a very accurate thermometer for measuring room temperature, but you try to use it to measure the temperature of lava; the thermometer is accurate, but it is not fit for purpose. This is the essence of data quality: it is not mere technical correctness, but contextual appropriateness.

But data quality alone is not enough. There is another layer above it that is equally important: reproducibility. Reproducibility is the ability to reconstruct the same result using the same data, the same tools, and the same steps, by another researcher or by the same researcher after time has passed. Reproducibility is not perfectionism; it is the cornerstone of the scientific method. If another researcher cannot repeat the project’s analysis and obtain the same result, the project’s conclusion remains an unproven claim, no matter how convincing it may appear. In the era of big data, where decisions that can change human lives depend on complex analyses, reproducibility has become an ethical and practical necessity, not merely academic curiosity.

جودة البيانات ليست مفهوماً فضفاضاً أو رفاهية أكاديمية. هي ببساطة "الدرجة التي تكون فيها البيانات مناسبة للاستخدام المقصود منها". قد تكون البيانات دقيقة تماماً لغرض معين، لكنها غير مناسبة لغرض آخر. تخيل أن يوجد في المثال ميزان حرارة دقيقاً جداً لقياس درجة حرارة غرفة، لكنك تحاول استخدامه لقياس درجة حرارة الحمم البركانية؛ الميزان دقيق، لكنه غير مناسب. هذا هو جوهر جودة البيانات: ليست مجرد صحة تقنية، بل مناسبة للسياق.

How do you achieve data quality and reproducibility? The answer lies in three fundamental pillars: provenance, versioning, and audit trail. Together, these three pillars form what can be called the "digital identity" of the data, answering three questions that any researcher must ask before using any data: Where did these data come from? How was it generated and validated? Can its source be traced at any tim?e.

لكن جودة البيانات وحدها لا تكفي. هناك طبقة أخرى فوقها لا تقل أهمية، وهي إمكانية إعادة الإنتاج. إعادة الإنتاج هي القدرة على إعادة بناء نفس النتيجة باستخدام نفس البيانات ونفس الأدوات ونفس الخطوات، من قبل باحث آخر أو من قبل نفس الباحث بعد مرور الوقت. إعادة الإنتاج ليست كمالية؛ إنها حجر الزاوية في المنهج العلمي. إذا لم يستطع باحث آخر تكرار تحليلك والحصول على النتيجة نفسها، فإن استنتاجك يظل مجرد ادعاء غير مثبت، مهما بدا مقنعاً.

في عصر البيانات الضخمة، حيث تعتمد قرارات قد تغير حياة البشر على تحليلات معقدة، أصبحت إعادة الإنتاج ضرورة أخلاقية وعملية، وليس مجرد فضول أكاديمي.

Let us start with the first pillar: provenance. Provenance is the complete record of the data's history. Not just a line saying "from database X," but a detailed document telling the story of the data from its birth in the laboratory to the moment it reaches the project’s analysis file. Good provenance is like a detailed birth certificate: it tells you the organism from which the sample was taken, the tissue from which it was extracted, the technology used for sequencing, the exact date of extraction, the laboratory that produced it, storage conditions, and any initial processing applied before it reached the project’s file. Imagine trying to solve a mystery with an artifact in front of you with no information about its origin; it could be valuable, or it could be a forgery. Provenance is what transforms data from a mysterious artifact into reliable evidence.

كيف تحقق جودة البيانات وإمكانية إعادة الإنتاج؟ الإجابة تكمن في ثلاث ركائز أساسية، هي المنشأ، وإدارة الإصدارات، وسجل التدقيق. هذه الركائز الثلاث تشكل معاً ما يمكن تسميته "الهوية الرقمية" للبيانات، وهي تجيب على ثلاثة أسئلة يجب أن يطرحها أي باحث قبل استخدام أي بيانات: من أين جاءت هذه البيانات؟ كيف أُنتجت وتم التحقق من صحتها؟ وهل يمكن تتبع مصدرها في أي وقت؟

The second pillar is versioning. In the world of bioinformatics, everything is constantly changing. Databases are updated daily, new versions of software tools appear continuously, and errors in old sequences are discovered and corrected. The same sequence may be stored in a database under a single accession number, but it may have undergone dozens of corrections over the years. If you use the old version without knowing, the workflow is building the project’s analysis on a potentially flawed foundation. Versioning means meticulously recording which version of any tool or database you used. Never say "I used an alignment tool," but say "I used BLAST version 2.12.0, with RefSeq database version 2025-01-15." This level of precision is what allows others to reconstruct the project’s software environment exactly as it was.

لنبدأ بالركيزة الأولى: المنشأ. المنشأ هو السجل الكامل لتاريخ البيانات. ليس مجرد سطر يقول "من قاعدة البيانات الفلانية"، بل هو وثيقة تفصيلية تروي قصة البيانات من لحظة ولادتها في المختبر إلى لحظة وصولها إلى ملف تحليلك. المنشأ الجيد يشبه شهادة ميلاد مفصلة: يخبرك بالكائن الحي الذي أُخذت منه العينة، والنسيج الذي استُخرجت منه، والتقنية التي استُخدمت في التسلسل، والتاريخ الدقيق للاستخراج، والمعمل الذي أنتجها، وظروف التخزين، وأي

معالجة أولية طُبقت عليها قبل أن تصل إلى ملفك. تخيل أن تحاول حل لغز، وأمامك قطعة أثرية دون أي معلومات عن مصدرها؛ قد تكون ثمينة، وقد تكون مزيفة. المنشأ هو ما يحول البيانات من أثر غامض إلى دليل موثوق.

The third pillar is the audit trail. An audit trail is a detailed chronological record of every action taken on the data: who opened the file, when, what changes were made, who made them, and why. This may seem like a small detail, but it becomes vital when you discover an error in the project's analysis weeks or months later. The audit trail allows you to go back step by step to pinpoint exactly when the error occurred. In sensitive systems like clinical diagnostics or pharmaceutical research, an audit trail is not optional; it is a legal and ethical requirement. It ensures that data has not been modified without knowledge, and that every decision can be traced back to its maker.

الركيزة الثانية هي إدارة الإصدارات. في عالم المعلوماتية الحيوية، كل شيء يتغير باستمرار. قواعد البيانات تتحديث يومياً، الإصدارات الجديدة من الأدوات البرمجية تظهر باستمرار، والأخطاء في التسلسلات القديمة تُكتشف وتُصحح. قد يكون التسلسل نفسه مخزناً في قاعدة البيانات تحت رقم انضمام واحد، لكنه قد خضع لعشرات التصحيحات عبر السنين. إذا استخدمت الإصدار القديم دون أن تدري، فأنت تبني تحليلك على أساس قد يكون خاطئاً. إدارة الإصدارات تعني أن تسجل بدقة أي إصدار من أي أداة أو قاعدة بيانات استخدمتها. لا تقل أبداً "استخدمت أداة المحاذاة"، بل قل "استخدمت أداة BLAST الإصدار 2.12.0، مع قاعدة بيانات RefSeq الإصدار 2025-01-15". هذا المستوى من الدقة هو ما يسمح للآخرين بإعادة بناء بيئتك البرمجية تماماً كما كانت.

الركيزة الثالثة هي سجل التدقيق. سجل التدقيق هو سجل زمني مفصل لكل إجراء تم على البيانات. من فتح الملف، ومتى، وما التغييرات التي أُجريت، ومن قام بها، ولماذا. قد يبدو هذا تفصيلاً صغيراً، لكنه يصبح حيوياً عندما تكتشف خطأ في تحليلك بعد أسابيع أو شهور. سجل التدقيق يسمح لك بالعودة خطوة بخطوة لتحديد متى حدث الخطأ بالضبط. في الأنظمة الحساسة مثل التشخيص السريري أو الأبحاث الدوائية، سجل التدقيق ليس خياراً، بل هو شرط قانوني وأخلاقي. فهو يضمن أن البيانات لم تُعدّل دون علم، وأن كل قرار يمكن تتبعه إلى صانعه.

هذه الركائز الثلاث ليست نظرية فقط؛ لها تطبيقات عملية يومية. عندما تبدأ أي مشروع معلوماتي حيوي، يجب أن تنشئ مجلداً منظمًا يحتوي على:

· ملف نصي يشرح مصدر كل ملف بيانات، مع روابط أو أرقام انضمام دقيقة.

· ملف إصدارات (مثل environment.yml أو requirements.txt) يوثق كل إصدارات الأدوات والمكتبات التي استخدمتها.

These three pillars are not merely theoretical; they have daily practical applications. When you start any bioinformatics project, it is appropriate to create an organized folder containing:

· سجل أوامر (log file) يسجل كل أمر كتبته في الطرفية، مع التاريخ والوقت.

· A text file explaining the source of each data file, with precise links or accession numbers.

· دفتر ملاحظات رقمي (مثل Jupyter Notebook) يشرح كل خطوة في تحليلك، مع التعليقات التي تشرح لماذا اخترت كل معلمة.

· A version file (e.g., environment.yml or requirements.txt) documenting every version of every tool and library you used.

· A command log file recording every command you typed in the terminal, with date and time.

· A digital notebook (e.g., Jupyter Notebook) explaining every step of the project's analysis, with comments explaining why you chose each parameter.

Let us pause for a moment at the concept of "data quality" itself. Data quality is not just a general idea; it can be measured and assessed by specific standards. In sequencing data, one of the most important quality metrics is the Base Quality Score, often known as the Phred score. This score gives you the probability that the base read by the sequencer is correct. A Phred score of 20 means there is a one in one hundred chance that the base is wrong. A Phred score of 30 means a one in one thousand chance. Imagine building an analysis on millions of bases, some of which may be wrong. Without knowing these scores, the workflow is building on unstable ground. Quality Control (QC) tools examine these scores and tell you if the data falls within acceptable limits, or if there is a problem requiring cleaning or even re-sequencing.

ودعنا نتوقف لحظة عند مفهوم "جودة البيانات" نفسه. جودة البيانات ليست مجرد فكرة عامة، بل يمكن قياسها وتقييمها بمعايير محددة. في بيانات التسلسل، واحدة من أهم مقاييس الجودة هي درجة جودة القاعدة (Base Quality Score)، والتي تُعرف غالباً بـ Phred score. هذه الدرجة تعطيك احتمالية أن تكون القاعدة التي قرأها جهاز التسلسل صحيحة. درجة Phred 20 تعني أن هناك احتمالاً واحداً من مائة أن القاعدة خاطئة. درجة Phred 30 تعني احتمالاً واحداً من ألف. للتوضيح، يمكن تصور حالة يكون فيها تبني تحليلاً على ملايين القواعد، بعضها قد يكون خاطئاً. دون معرفة هذه الدرجات، أنت تبني على أرض غير مستقرة. فحوصات الجودة (Quality Control)

هي الأدوات التي تفحص هذه الدرجات وتخبرك إذا كانت البيانات ضمن الحدود المقبولة، أو إذا كانت هناك مشكلة تتطلب التنظيف أو حتى إعادة التسلسل.

But what if the data is of high quality, but it is not reproducible because you forgot to record a crucial step? This is where decision documentation comes in. In any analysis, you will encounter moments where the analysis requires to make a decision: which threshold to choose? Which normalization method to prefer? Do you remove outliers or keep them? These decisions are not neutral; each decision carries an assumption, and each assumption may change the result. If you do not document these decisions, no one will be able to know why you reached the project's result, not even you six months later. Good documentation is not merely a list of steps; it is a coherent narrative explaining "why" and not just "what.

And here we reach a golden rule no less important than any algorithm: If you did not document it, it did not happen. This rule may seem harsh, but it is true. In science, a claim without documented evidence has no value. Documentation is the proof that the analysis was performed, and that the results are not just an illusion or a passing error. Documentation protects the researcher from their own forgetfulness, and protects the scientific community from relying on unverifiable results.

لكن ماذا لو كانت البيانات ذات جودة عالية، لكنها غير قابلة لإعادة الإنتاج لأنك نسيت تسجيل خطوة حاسمة؟ هنا يأتي دور توثيق القرارات. في أي تحليل، ستواجه لحظات تحتاج فيها إلى اتخاذ قرار: أي عتبة تختار؟ أي طريقة تطبيع تفضل؟ هل تزيل القيم المتطرفة أم تحتفظ بها؟ هذه القرارات ليست محايدة؛ كل قرار يحمل افتراضاً، وكل افتراض قد يغير النتيجة. إذا لم توثق هذه القرارات، فلن يستطيع أحد معرفة لماذا وصلت إلى نتيجتك، ولا حتى أنت بعد ستة أشهر. التوثيق الجيد ليس مجرد قائمة خطوات، بل هو رواية متماسكة تشرح "لماذا" وليس فقط "ماذا".

To clarify further, for illustration, consider a case in which are conducting a complex genomic analysis. You start with a raw file containing millions of reads. You pass it through several tools: a quality control tool, a tool to trim low-quality parts, a tool to align reads to a reference genome, and a tool to calculate gene expression. At the end, you get a list of genes that differ in expression between a healthy and a diseased sample. This analysis may have taken days. But the question is: how do you know each step was performed correctly? How do you know the second tool did not change something in the data without the project's knowledge? How do you know the threshold you chose in the fourth step was appropriate for this particular data? The answer: through documentation. Every step must have its parameters and outputs recorded, every intermediate file must

be saved, and every decision must be explained. This way, the workflow can go back to any point in the analysis and review it, and others can do the same.

وهنا نصل إلى قاعدة ذهبية لا تقل أهمية عن أي خوارزمية: إذا لم تُوثقه، لم يحدث. قد تكون هذه القاعدة قاسية، لكنها حقيقية. في العلوم، الادعاء بدون دليل موثق لا قيمة له. التوثيق هو الدليل على أن التحليل قد تم، وأن النتائج ليست مجرد وهم أو خطأ عابر. التوثيق يحمي الباحث من نسيانه هو نفسه، ويحمي المجتمع العلمي من الاعتماد على نتائج لا يمكن التحقق منها.

Reproducibility does not merely mean re-running the code. It means that any other researcher, regardless of their location or time, can understand what you did, can access the same data (or its equivalent), can apply the same steps, and obtain the same result. This is the gold standard. If the project's analysis depends on a local file on the project's computer, or on a database whose version you did not document, or on a software tool whose version you did not specify, then the project's analysis is not reproducible, no matter how exciting its results may be.

للتوضيح أكثر، للتوضيح، يمكن تصور حالة يكون فيها تجري تحليلاً جينياً معقداً. تبدأ بملف خام يحتوي على ملايين القراءات. تمررها عبر عدة أدوات: أداة لفحص الجودة، أداة لقص الأجزاء منخفضة الجودة، أداة لمحاذاة القراءات على الجينوم المرجعي، وأداة لحساب التعبير الجيني. في النهاية، تحصل على قائمة بالجينات التي يختلف تعبيرها بين عينة سليمة وأخرى مريضة. هذا التحليل قد استغرق أياماً. لكن السؤال هو: كيف تعرف أن كل خطوة تمت بشكل صحيح؟ كيف تعرف أن الأداة الثانية لم تغير شيئاً في البيانات دون أن تدري؟ كيف تعرف أن العتبة التي اخترتها في الخطوة الرابعة كانت مناسبة لهذه البيانات تحديداً؟ الإجابة: من خلال التوثيق. كل خطوة يجب أن تسجل معلماتها ومخرجاتها، وكل ملف وسيط يجب حفظه، وكل قرار يجب شرحه. بهذه الطريقة، يمكن العودة إلى أي نقطة في التحليل وتفقدها، ويمكن لآخرين أن يفعلوا الشيء نفسه.

In an ideal world, every bioinformatics analysis should be accompanied by three things: the raw data as it came from the sequencer, the complete code with all versions, and a report documenting every decision, parameter, and transformation. This triad (data, code, documentation) is what makes science auditable and buildable. Without any of them, the analysis remains suspended in the air, neither trustworthy nor developable.

قابلية إعادة الإنتاج لا تعني فقط إعادة تشغيل الكود. هي تعني أن أي باحث آخر، بغض النظر عن مكانه أو زمانه، يمكنه فهم ما فعلته، ويمكنه الوصول إلى نفس البيانات (أو ما يعادلها)، ويمكنه تطبيق نفس الخطوات، والحصول على نفس النتيجة. هذه هي المعيار الذهبي. إذا كان تحليلك يعتمد على ملف محلي على حاسوبك، أو على قاعدة بيانات لم توثق إصدارها، أو على أداة برمجية لم تحدد إصدارها، فتحليلك غير قابل لإعادة الإنتاج، مهما كانت نتائجه مثيرة.

The final conclusion of this chapter is that data quality and reproducibility are not an additional step at the end of a project; they are an integral part of the analysis process itself. They begin before data collection, through careful experimental design and the definition of quality criteria. They continue during the analysis, through continuous documentation and version recording. And they end after the analysis, through sharing data, code, and documentation with the scientific community. A good researcher is not just one who uses tools skillfully, but one who builds their work on a foundation that others can build upon in turn.

في عالم مثالي، كل تحليل معلوماتي حيوي يجب أن يُرفق بثلاثة أشياء: البيانات الخام كما خرجت من جهاز التسلسل، الكود المستخدم بالكامل مع جميع الإصدارات، وتقرير يوثق كل قرار وكل معلمة وكل تحويل. هذه الثلاثية (البيانات، الكود، التوثيق) هي ما يجعل العلم قابلاً للتدقيق والبناء عليه. بدون أي منها، يظل التحليل معلقاً في الهواء، لا يمكن الوثوق به ولا يمكن تطويره.

الخلاصة النهائية لهذا الفصل هي أن جودة البيانات وإمكانية إعادة الإنتاج ليستا خطوة إضافية في نهاية المشروع، بل هي جزء لا يتجزأ من عملية التحليل نفسها. تبدأ قبل جمع البيانات، من خلال تصميم تجربة محكمة وتحديد معايير الجودة. تستمر أثناء التحليل، من خلال التوثيق المستمر وتسجيل الإصدارات. وتنتهي بعد التحليل، من خلال مشاركة البيانات والكود والتوثيق مع المجتمع العلمي. الباحث الجيد ليس فقط من يستخدم الأدوات بمهارة، بل من يبني عمله على أساس يمكن للآخرين أن يبنوا عليه بدورهم.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Data Quality and Reproducibility" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "جودة البيانات وإمكانية إعادة الإنتاج" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "well-documented data always guarantees correct results." This is incorrect, because good documentation makes errors visible and traceable, but it does not prevent them. The correct statement is: "well-documented data makes results auditable and reproducible, and allows others to evaluate the validity of the conclusions themselves." Documentation is a tool for transparency, not a guarantee of correctness.

مثال على التجاوز: القول إن "البيانات الموثقة جيداً تضمن نتائج صحيحة دائماً". هذا غير صحيح، لأن التوثيق الجيد يجعل الأخطاء مرئية وقابلة للتتبع، لكنه لا يمنعها. الصحيح هو: "البيانات الموثقة جيداً تجعل النتائج قابلة للتدقيق

وإعادة الإنتاج، وتسمح للآخرين بتقييم صحة الاستنتاجات بأنفسهم." التوثيق هو أداة للشفافية، وليس ضماناً للصواب.

Example: The presence of a data file with complete documentation does not mean that the biological conclusion drawn from it is correct. Documentation tells us how the data was produced, but it does not prove that the data represents a real biological phenomenon. Claiming that "thi.

مثال: وجود ملف بيانات مع توثيق كامل لا يعني أن الاستنتاج البيولوجي المستخلص منه صحيح. التوثيق يخبرنا كيف أُنتجت البيانات، لكنه لا يثبت أن البيانات تمثل ظاهرة بيولوجية حقيقية. الادعاء بأن "هذه البيانات تثبت وجود علاقة بين جين X ومرض Y" يحتاج إلى أدلة إحصائية وتجريبية إضافية، وليس فقط إلى توثيق جيد للملف. الفصل بين ما يخبرنا به التوثيق (وهو تاريخ البيانات وتحويلاتها) وما نستنتجه عن البيولوجيا (وهو تفسير وظيفي أو سببي) هو جوهر التفكير النقدي في المعلوماتية الحيوية.

Part 2 — Foundation: Preparing the Working Environment

الباب 2 — مرحلة التأسيس: تجهيز بيئة العمل

Chapter 4 — Installing Python and Understanding PATH

الفصل 4 — تثبيت Python وفهم PATH

الفكرة المركزية: المفسر، PATH، الإصدارات، البيئة كجزء من التجربة الحاسوبية.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

تثبيت Python يضع المفسر على الجهاز، لكنه لا يجيب وحده عن سؤال أي مفسر سيُستخدم فعلياً. PATH قد يجعل الأمر نفسه يشير إلى تثبيت مختلف، ولهذا فإن تسجيل الإصدار والمسار جزء من سجل التجربة. هذه التفاصيل تبدو تشغيلية، لكنها تصبح علمية عندما يكون اختلاف البيئة سبباً لاختلاف النتيجة.

Installing Python places an interpreter on the machine, but it does not by itself tell us which interpreter will actually be invoked. PATH can make the same command point to a different installation, so recording the version and executable path belongs in the experimental record. These details look operational, but become scientific when environmental differences change results.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is the Interprete?r

1. ما هو المفسر (Interpreter)؟

Python is an interpreted language, not a compiled one (like C++). This means that the code you write does not directly translate into machine language; rather, it is read line by line by another program called the "interpreter." The interpreter is the engine that executes the project's code. For illustration, consider a case in which are writing a message in a foreign language, and the interpreter is the simultaneous translator who converts the project's message into a language the computer understands. If the translator changes, the meaning may change or the translation may fail entirely.

Python هي لغة مفسرة، وليست لغة مترجمة (مثل C++). هذا يعني أن الكود الذي تكتبه لا يتحول إلى لغة الآلة مباشرة، بل يُقرأ سطراً سطراً بواسطة برنامج آخر يُدعى "المفسر". المفسر هو المحرك الذي ينفذ كودك. للتوضيح، يمكن تصور حالة يكون فيها تكتب رسالة بلغة أجنبية، والمفسر هو المترجم الفوري الذي يحول رسالتك إلى لغة يفهمها الحاسوب. إذا تغير المترجم، قد يتغير المعنى أو قد تفشل الترجمة تماماً.

Python versions are not identical. Code that runs on Python 3.8 may not run on Python 3.11 because some functions have changed or been removed, or because the default behavior of some operations has changed. Therefore, knowing the exact version number of the interpreter you used is not a minor detail; it is an essential part of describing the project's experiment. If you write in the project's research paper "I used Python," that is not sufficient. The correct statement is "I used Python 3.11.4.".

إصدارات Python ليست متطابقة. الكود الذي يعمل على Python 3.8 قد لا يعمل على Python 3.11 لأن بعض الدوال تغيرت أو أُزيلت، أو لأن السلوك الافتراضي لبعض العمليات اختلف. لذلك، معرفة رقم الإصدار الدقيق للمفسر الذي استخدمته ليست تفصيلاً ثانوياً؛ إنها جزء أساسي من وصف تجربتك. إذا كتبت في ورقتك البحثية "استخدمت Python"، فهذا غير كافٍ. الصحيح أن تكتب "استخدمت Python 3.11.4".

2. What Is PATH؟

2. ما هو PATH؟

PATH is an environment variable in the operating system. the workflow can imagine it as a list of addresses that the computer searches to find the programs you request. When you type the command python in the terminal, the computer does not directly know where this program is. It searches through all the folders listed in the PATH list, and as soon as it finds a file named python (or python.exe on Windows) in one of these folders, it executes it.

PATH هو متغير بيئي (Environment Variable) في نظام التشغيل. يمكن تخيله كقائمة عناوين يبحث فيها الحاسوب عن البرامج التي تطلبها. عندما تكتب الأمر python في الطرفية، لا يعرف الحاسوب مكان هذا البرنامج مباشرة. هو يبحث في جميع المجلدات المذكورة في قائمة PATH، وبمجرد أن يجد ملفاً اسمه python (أو python.exe في Windows) في أحد هذه المجلدات، ينفذه.

The problem arises when the workflow has more than one Python installation on the project's system. You might have an older version installed in /usr/bin/python and a newer version in /usr/local/bin/python. The order in which folders appear in the PATH determines which version will be used when you type python. If the older folder appears before the newer one in the list, you will run the older version without knowing it. This is the source of a very common error, where the researcher says, "The code works on my machine but not on my colleague's." The reason is often a difference in PATH order or version differences.

المشكلة تظهر عندما يكون يوجد في المثال أكثر من تثبيت لـ Python على نظامك. قد يكون يوجد في المثال إصدار قديم مثبت في مجلد /usr/bin/python، وإصدار جديد مثبت في /usr/local/bin/python. الترتيب الذي تظهر به المجلدات في PATH يحدد أي إصدار سيُستخدم عندما تكتب python. إذا كان المجلد القديم يظهر قبل الجديد في القائمة، فستشغل الإصدار القديم دون أن تدري. هذا هو مصدر أخطاء شائعة جداً، حيث يقول الباحث "الكود يعمل على جهازي لكنه لا يعمل على جهاز زميلي". السبب غالباً هو اختلاف ترتيب PATH أو اختلاف الإصدارات.

3. Verifying the Active Interpreter

3. التحقق من المفسر النشط

Therefore, the first practical step in any project is to verify which interpreter the workflow is actually using. Do not merely install Python and assume that the command points to the correct version. Use the following commands to check.

لذلك، أول خطوة عملية في أي مشروع هي التحقق من المفسر الذي تستخدمه فعلياً. لا تكتفِ بتثبيت Python والظن أن الأمر يشير إلى الإصدار الصحيح. استخدم الأوامر التالية لتتحقق:

Language / نوع الكود: **Bash / Shell**

```
python --version

which python   # على Linux/macOS

where python   # على Windows
```

On some Unix systems, the command python may not point to Python 3 by default; you may need to use python3 --version instead. This difference is not a system flaw; it is a reminder that precision in documenting the full command name is part of reproducibility. Always record the full command you used, not just the program name.

4. The Environment as a Measuring Instrument

في بعض أنظمة Unix، قد لا يستجيب الأمر python لإصدار Python 3 افتراضياً، بل قد تحتاج إلى استخدام python3 --version. هذا الاختلاف ليس عيباً في النظام، بل هو تذكير بأن الدقة في توثيق الاسم الكامل للأمر هي جزء من إعادة الإنتاج. سجل دائماً الأمر الكامل الذي استخدمته، ليس فقط اسم البرنامج.

In a biological laboratory, you would never accept a measurement from an uncalibrated instrument. Before using a thermometer, you ensure it gives the correct reading at a known temperature (such as the boiling point of water). Similarly, a computational result should not be interpreted without knowing the environment in which it was executed. The interpreter, its versions, and package locations are the "measuring instruments" in the programming world. If you do not know their calibration, you do not know whether the result reflects correct code behavior or the behavior of a specific environment.

4. البيئة كأداة قياس

For illustration, consider a case in which conduct a complex analysis and find an exciting result. You publish it in a research paper. A year later, another team tries to replicate the project's experiment, but they get a different result. After weeks of investigation, they discover they used a newer version of Python where the behavior of a specific function had changed. This scenario is not fictional; it happens every day in reality. The difference between a professional researcher and an amateur is that the professional records the interpreter version before starting the analysis, not after discovering the problem.

في المختبر البيولوجي، لا تقبل أي قياس من جهاز غير معروف المعايرة. قبل أن تستخدم ميزاناً حرارة، تتأكد من أنه يُعطي القراءة الصحيحة عند درجة حرارة معروفة (مثل نقطة غليان الماء). وبالمثل، لا ينبغي تفسير نتيجة برمجية من دون معرفة البيئة التي نُفذت فيها. المفسر وإصداراته ومواقع الحزم هي "أجهزة القياس" في عالم البرمجة. إذا لم تعرف معايرتها، فأنت لا تعرف إن كانت النتيجة تعكس سلوك الكود الصحيح أم سلوك بيئة معينة.

5. Separating Environment Problems from Program Problems

للتوضيح، يمكن تصور حالة يكون فيها تجري تحليلاً معقداً، وتجد نتيجة مثيرة. تنشرها في ورقة بحثية. بعد عام، يحاول فريق آخر تكرار تجربتك، لكنهم يحصلون على نتيجة مختلفة. بعد أسابيع من البحث، يكتشفون أنهم استخدموا إصداراً أحدث من Python حيث تغيرت طريقة عمل دالة معينة. هذا الموقف ليس خيالياً؛ إنه يحدث في الواقع كل يوم. الفرق بين الباحث المحترف والآخر هو أن المحترف يسجل إصدار المفسر قبل أن يبدأ التحليل، وليس بعد أن يكتشف المشكلة.

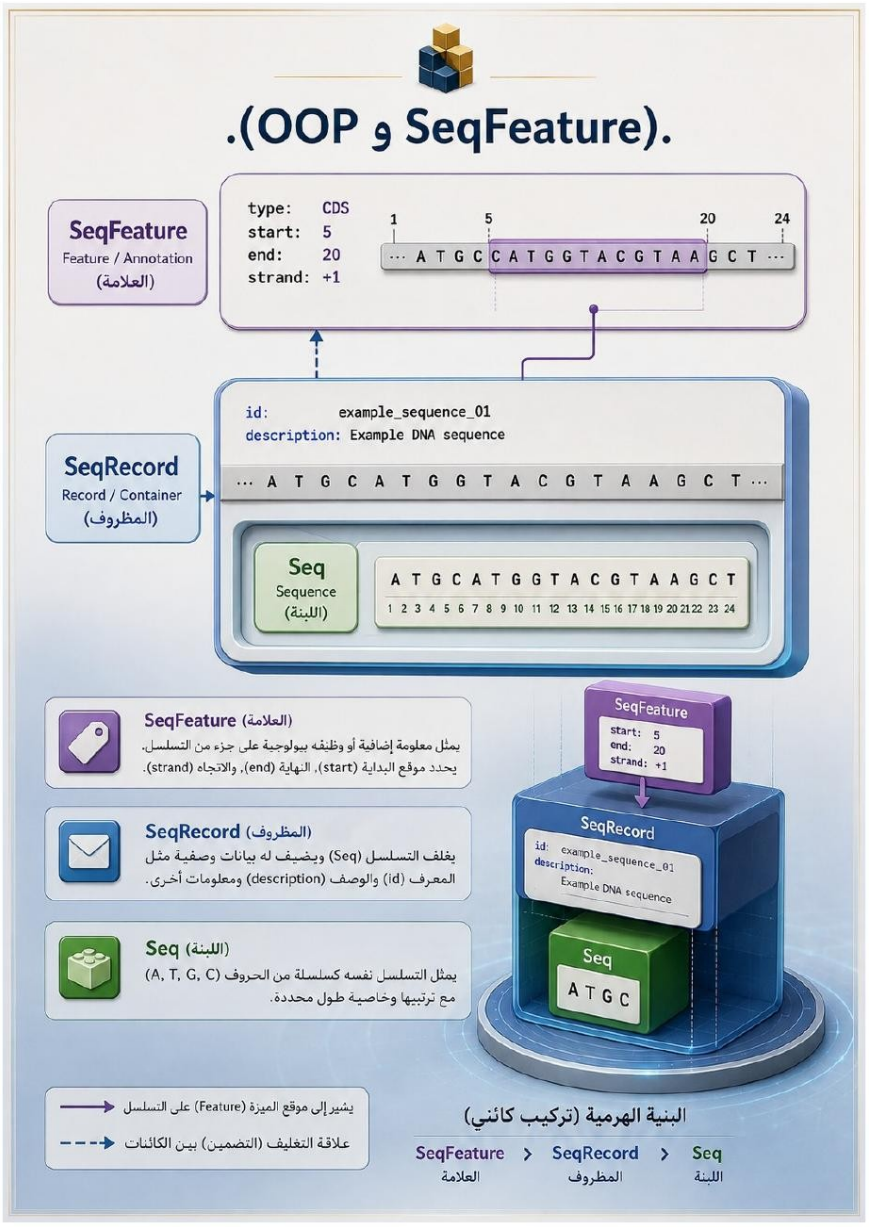
When an error occurs in the project's code, there are two main possible causes. The first: the code itself contains a logical error (bug). The second: the environment is different from what the code expects (e.g., a different version, a missing library, or an incorrect path). If the workflow is not sure about the environment the workflow is working in, you will not know where to look for the error. Verifying the interpreter and PATH is not merely a formality; it is a

diagnostic tool that helps you pinpoint the source of the problem. When you know exactly which Python version and which path the project's system uses, you turn the error from a confusing puzzle into a specific, solvable problem.

5. الفصل بين مشاكل البيئة ومشاكل البرنامج

This is where the concept of "isolation," which we will explore in the next chapter, comes in. Virtual environments allow you to freeze specific versions so they do not change, reducing PATH and version issues. But before you reach virtual environments, you must understand the foundation: the interpreter and PATH are the pillars upon which every software environment stands.

عندما يحدث خطأ في كودك، هناك سببان محتملان رئيسيان. الأول: أن الكود نفسه يحتوي على خطأ منطقي (bug). الثاني: أن البيئة مختلفة عما يتوقعه الكود (مثل إصدار مختلف، أو مكتبة مفقودة، أو مسار خاطئ). إذا لم تكن متأكداً من البيئة التي تعمل فيها، فلن تعرف أين تبحث عن الخطأ. التحقق من المفسر و PATH ليس مجرد إجراء شكلي؛ هو أداة تشخيصية تساعدك على تحديد مصدر المشكلة. عندما تعرف بالضبط أي إصدار من Python وأي مسار يستخدمه نظامك، فإنك تحول الخطأ من لغز محير إلى مشكلة محددة يمكن حلها.


.(OOP و SeqFeature)
SeqFeature
Feature / Annotation
(العلامة)
type: CDS
start: 5
end: 20
strand: +1
1
5
20
24
··· A T G C C A T G G T A C G T A A G C T ···
SeqRecord
Record / Container
(المظروف)
id: example_sequence_01
description: Example DNA sequence
··· A T G C A T G G T A C G T A A G C T ···
Seq
Sequence
(اللبنة)
A T G C A T G G T A C G T A A G C T
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
SeqFeature (العلامة)
يمثل معلومة إضافية أو وظيفه بيولوجية على جزء من التسلسل.
يحدد موقع البداية (start)، النهاية (end)، والاتجاه (strand).
SeqRecord (المظروف)
يغلف التسلسل (Seq) ويضيف له بيانات وصفية مثل
المعرف (id) والوصف (description) ومعلومات أخرى.
Seq (اللبنة)
يمثل التسلسل نفسه كسلسلة من الحروف (A, T, G, C)
مع ترتيبها وخاصية طول محددة.
SeqFeature
start: 5
end: 20
strand: +1
SeqRecord
id: example_sequence_01
description:
Example DNA sequence
Seq
A T G C
يشير إلى موقع الميزة (Feature) على التسلسل
علاقة التغليف (التضمين) بين الكائنات
البنية الهرمية (تركيب كائني)
SeqFeature
العلامة
SeqRecord
المظروف
Seq
اللبنة

Infographic 26 / إنفوجرافيك 26

6. Practical Application

وهنا يأتي دور مفهوم "العزلة" الذي سنتعمق فيه في الفصل التالي. البيئة الافتراضية تسمح لك بتجميد إصدارات معينة بحيث لا تتغير، مما يقلل من مشاكل PATH والإصدارات. لكن قبل أن تصل إلى البيئات الافتراضية، يجب أن تفهم الأساس: المفسر و PATH هما الركيزتان اللتان تقوم عليهما كل بيئة برمجية.

6. التطبيق العملي

Before you start any bioinformatics project, open the terminal, run the following commands, and record the output in a digital notebook (such as a text file or Jupyter Notebook).

قبل أن تبدأ أي مشروع معلوماتي حيوي، افتح الطرفية ونفّذ الأوامر التالية، وسجل الناتج في مفكرة رقمية (مثل ملف نصي أو Jupyter Notebook):

Language / نوع الكود: **Bash / Shell**

```
# سجل إصدار Python
python --version
# سجل المسار الكامل للمفسر
which python   # أو where python على Windows
# سجل محتويات PATH (لفهم ترتيب البحث)
echo $PATH     # على Linux/macOS
echo %PATH%    # على Windows
```

This information is not merely administrative; it is part of the description of the project's experiment. When you give the project's file to a colleague or publish it with the project's research paper, this information will help them reconstruct the project's environment precisely. And this is the essence of reproducibility.

هذه المعلومات ليست مجرد سجل إداري؛ إنها جزء من وصف تجربتك. عندما تعطي ملفك لزميل أو تنشره مع ورقتك البحثية، ستساعده هذه المعلومات على إعادة بناء بيئتك بدقة. وهذا هو جوهر إعادة الإنتاج.

للتأكد من أن كل شيء يعمل، اكتب برنامجاً بسيطاً في ملف جديد اسمه test.py بالمحتوى التالي:

```
print("Hello, World!")
```

ثم شغله من الطرفية:

```
python test.py
```

To make sure everything works, write a simple program in a new file named test.py with the following content:

إذا رأيت النص Hello، World! على الشاشة، فبيئتك تعمل بشكل صحيح. هذا الاختبار البسيط هو "فحص العقل" للبرمجة: يتأكد أن المفسر يستطيع قراءة ملف وتنفيذ أمر أساسي. إنه مثل قياس درجة حرارة الغرفة قبل بدء تجربة حساسة.

```
print("Hello, World!")
```

Then run it from the terminal:

```
python test.py
If you see Hello, World! on the screen, the project’s environment is working correctly. This simple test is the "sanity check" of programming: it confirms that the interpreter can read a file and execute a basic command. It is like measuring room temperature before starting a sensitive experiment.
```

Chapter Summary

خلاصة الفصل

Installing Python and understanding PATH are not merely technical steps to start programming. They are the first steps of scientific thinking about the software environment. The interpreter is the tool that executes the project’s code, and PATH is the map by which the system finds this tool. Knowing the interpreter's version and location is as important as knowing the calibration of a thermometer in a laboratory. It is the difference between a result that can be trusted and a result that might be merely an illusion caused by a different environment. The next chapter will build on this foundation and introduce virtual environments as a tool to isolate the project’s project's dependencies and ensure their consistency.

تثبيت Python وفهم PATH ليسا مجرد خطوات تقنية لبدء البرمجة. هما أولى خطوات التفكير العلمي في البيئة البرمجية. المفسر هو الأداة التي تنفذ كودك، و PATH هو الخريطة التي يجد بها النظام هذه الأداة. معرفة إصدار المفسر وموقعه لا تقل أهمية عن معرفة معايرة ميزان حرارة في مختبر. هي الفرق بين نتيجة يمكن الوثوق بها ونتيجة قد تكون مجرد وهم ناتج عن بيئة مختلفة. الفصل القادم سيبني على هذا الأساس، ويقدم لك البيئات الافتراضية كأداة لعزل اعتماديات مشروعك وضمان اتساقها.

Five questions that keep the scientific reasoning explicit.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Installing Python and Understanding PATH" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "تثبيت Python وفهم PATH" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "installing Python means the code will run the same way on any machine." This is incorrect, because Python versions differ, and PATH order differs between systems. The correct statement is:

"Installing Python is the first step, but knowing which version actually responds when you run the command, and which path the system uses, is what ensures reproducibility." A simple test: before starting any project, run python --version and which python and record the output. If you cannot reproduce this output a year later, the project’s environment is not documented.

مثال على التجاوز: القول إن "تثبيت Python يعني أن الكود سيعمل بنفس الطريقة على أي جهاز". هذا غير صحيح، لأن إصدارات Python تختلف، وترتيب PATH يختلف بين الأنظمة. الصحيح هو: "تثبيت Python هو الخطوة الأولى، لكن معرفة أي إصدار يستجيب فعلياً عند تشغيل الأمر، وأي مسار يستخدمه النظام، هي ما يضمن تكرارية النتائج." اختبار بسيط: قبل أن تبدأ أي مشروع، شغّل python --version و which python وسجل الناتج. إذا لم تستطع إعادة هذا الناتج بعد عام، فبيئتك غير موثقة.

Example: Knowing that the workflow is using Python 3.11 is a computational representation of the project’s environment. But claiming that "the results you obtained reflect a real biological phenomenon" is not proven by the interpreter version. The Python version tells us about the technical environment, not the validity of the biological conclusion. Practical test: if you find an exciting result, Practical check: "Does this result depend on the behavior of a specific Python version, or does it reflect a real biological phenomenon that can be confirmed by independent evidenc?"e.

مثال: معرفة أنك تستخدم Python 3.11 هو تمثيل حسابي لبيئتك. لكن الادعاء بأن "النتائج التي حصلت عليها تعكس سلوكاً بيولوجياً حقيقياً" لا يثبته إصدار المفسر. إصدار Python يخبرنا عن البيئة التقنية، وليس عن صحة الاستنتاج البيولوجي. اختبار عملي: إذا وجدت نتيجة مثيرة، فحص تطبيقي: "هل هذه النتيجة تعتمد على سلوك إصدار معين من Python، أم أنها تعكس ظاهرة بيولوجية حقيقية يمكن تأكيدها بأدلة مستقلة؟".

مثال تطبيقي على أمر التحقق من البيئة:

· المدخلات: أمر python --version في الطرفية.

· الافتراضات: أن الأمر python يشير إلى الإصدار الذي تعتقد أنه يشير إليه، وأن PATH لم يتغير منذ آخر استخدام.

· المخرجات: رقم الإصدار الدقيق (مثل Python 3.11.4).

· القيود: هذا الأمر لا يخبرك عن إصدارات المكتبات المثبتة، ولا عن موقع الحزم، ولا عن المتغيرات البيئية الأخرى التي قد تؤثر على التنفيذ.

Applied example of the environment verification command.

اختبار: قبل أن تبدأ التحليل، اكتب قائمة واضحة بهذه العناصر الأربعة لكل أمر ستستخدمه. إذا لم تستطع تحديد أحدها، فأنت لا تعرف بيئتك بما يكفي.

· Inputs: the python --version command in the terminal.

· Assumptions: that the python command points to the version you think it does, and that PATH has not changed since the last use.

· Outputs: the exact version number (e.g., Python 3.11.4).

· Limitations: this command does not tell you about installed library versions, nor package locations, nor other environment variables that may affect execution.

Test: before starting the project's analysis, write a clear list of these four elements for every command you will use. If you cannot identify one of them, you do not know the project's environment well enough.

عند توثيق بيئتك، يجب تسجيل ما يلي في ملف نصي أو مفكرة رقمية قبل تشغيل أي كود:

· المصدر: نظام التشغيل (Windows، Linux، macOS) وإصداره.

· الإصدار: إصدار Python الدقيق (مثل 3.11.4)، وإصدار مدير الحزم (pip).

· المعلمات: أي متغيرات بيئية قمت بتعيينها (مثل PYTHONPATH).

· التحويلات: أي خطوات قمت بها لتثبيت أو تحديث الأدوات قبل التحليل.

When documenting the project's environment, record the following in a text file or digital notebook before running any code.

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة بناء نفس البيئة دون أن يسألني أي سؤال؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the operating system (Windows, Linux, macOS) and its version.

· Version: the exact Python version (e.g., 3.11.4), and the package manager version (pip).

· Parameters: any environment variables you set (e.g., PYTHONPATH).

· Transformations: any steps you took to install or update tools before analysis.

Test: Practical check: "If I gave this record to a colleague a year later, could they rebuild the same environment without asking me any questions?" If the answer is no, the documentation is insufficient.

قبل أن تبدأ تحليلك على بيانات حقيقية، اختبر بيئتك بأمر بسيط ومعروف النتيجة. اكتب سكربتاً صغيراً يقوم بطباعة "Hello، World!" أو حساب مجموع رقمين، وشغّله. إذا نجح، فأنت تعرف أن المفسر يعمل والمسار صحيح. إذا فشل، فستعرف أن المشكلة في البيئة (مثل PATH أو الإصدار) وليس في الكود المعقد. هذه العادة البسيطة توفر ساعات من البحث عن أخطاء في بيئات معقدة.

Before you start the project's analysis on real data, test the project's environment with a simple command with a known outcome. Write a small script that prints "Hello, World!" or calculates the sum of two numbers, and run it. If it succeeds, you know the interpreter works and the path is correct. If it fails, you know the problem is in the environment (such as PATH or version) and not in the complex code. This simple habit saves hours of searching for errors in complex environments.

Chapter 5 — Virtual Environments and Packages

الفصل 5 — البيئات الافتراضية والحزم

Deeper Conceptual Explanation

الفكرة المركزية: العزل، الاعتماديات، تثبيت الإصدارات، حدود reproducibility.

شرح أعمق للمفهوم

Software isolation prevents one project from silently changing another project's environment, but it does not make an entire system reproducible by itself. Operating-system details, reference data, live data sources, and human decisions can still alter results. A virtual environment is therefore one layer of reproducibility, not a guarantee by itself.

العزل البرمجي يمنع مشروعاً من تغيير بيئة مشروع آخر، لكنه لا يجعل النظام بأكمله قابلاً لإعادة الإنتاج تلقائياً. تبقى نسخة نظام التشغيل، والبيانات المرجعية، ومصادر البيانات الحية، وبعض القرارات البشرية عوامل قد تغير الناتج.

لذلك تُفهم البيئة الافتراضية كطبقة من طبقات إعادة الإنتاج، لا كضمان منفرد.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Why Virtual Environments?

1. لماذا البيئات الافتراضية؟

When you write code in Python, you rarely depend only on the standard library. You use external packages like Biopython, NumPy, or Pandas. These packages have

different versions, and each version may change function behavior, fix bugs, or add new features. The problem arises when the workflow has two projects on the same machine, one needing Biopython 1.79 and the other needing Biopython 1.81. If you install version 1.81 system-wide, the first project may stop working. And vice versa. This is a "dependency conflic.?t.

عندما تكتب كوداً في Python، نادراً ما تعتمد فقط على المكتبة القياسية (standard library). أنت تستخدم حزماً خارجية مثل NumPy، Biopython، أو Pandas. هذه الحزم لها إصدارات مختلفة، وكل إصدار قد يغير سلوك الدوال أو يصلح أخطاء أو يضيف ميزات جديدة. المشكلة تظهر عندما يكون يوجد في المثال مشروعان على نفس الجهاز، أحدهما يحتاج إلى Biopython 1.79 والآخر يحتاج إلى Biopython 1.81. إذا قمت بتثبيت الإصدار 1.81 على النظام الرئيسي، فقد يتوقف المشروع الأول عن العمل. والعكس صحيح. هذا هو "تعارض الاعتماديات" (dependency conflict).

A virtual environment solves this problem by creating an "isolated room" for each project. Inside this room, the workflow can install any version of any package without affecting the rest of the system or other projects. A virtual environment is a folder containing an independent copy of Python and a separate folder for packages. When you activate the environment, this folder becomes the default destination for installing packages, and the interpreter inside it becomes the active interpreter.

البيئة الافتراضية تحل هذه المشكلة عن طريق إنشاء "غرفة معزولة" لكل مشروع. داخل هذه الغرفة، يمكن تثبيت أي إصدار من أي حزمة دون التأثير على بقية النظام أو على مشاريع أخرى. البيئة الافتراضية هي مجلد يحتوي على نسخة مستقلة من Python ومجلد منفصل للحزم. عندما تُفعّل البيئة، يصبح هذا المجلد هو الوجهة الافتراضية لتثبيت الحزم، ويصبح المفسر الموجود فيها هو المفسر النشط.

2. How Does a Virtual Environment Work?

2. كيف تعمل البيئة الافتراضية؟

عندما تنشئ بيئة افتراضية باستخدام الأمر python -m venv .venv، يحدث الآتي:

· يُنشأ مجلد جديد (عادةً باسم .venv) يحتوي على نسخة من مفسر Python.

When you create a virtual environment using the command python -m venv .venv, the following happens.

· يُنشأ مجلد فرعي lib أو Lib يحتوي على مجلد الحزم الخاص بالبيئة.

· A new folder (usually named .venv) is created containing a copy of the Python interpreter.

· يُنشأ ملف تفعيل (activate) في مجلد bin (Unix) أو Scripts (Windows) يعدّل متغيرات البيئة مؤقتاً.

· A subfolder lib or Lib is created containing the environment's package folder.

· An activation file is created in the bin (Unix) or Scripts (Windows) folder that temporarily modifies environment variables.

يحدث التالي عندما تفعّل البيئة (source .venv/bin/activate أو .venv\Scripts\activate):

· يُضاف مسار المفسر في البيئة إلى بداية قائمة PATH، بحيث يصبح الأمر python يشير إلى مفسر البيئة وليس النظام.

When you activate the environment (source .venv/bin/activate or .venv\Scripts\activate), the following happens.

· يُضبط متغير VIRTUAL_ENV ليشير إلى مجلد البيئة.

· The path to the environment's interpreter is added to the beginning of the PATH list, so the python command points to the environment's interpreter, not the system's.

· يصبح الأمر pip مثبتاً في البيئة، وتُثبت الحزم في مجلد البيئة وليس في النظام.

· The VIRTUAL_ENV variable is set to point to the environment folder.

· The pip command is installed in the environment, and packages are installed in the environment's folder, not system-wide.

This isolation is temporary: as soon as you type deactivate, the variables return to their original state and the python command returns to the system interpreter. The environment does not change the system; it only changes how the system sees commands and packages during the current session.

هذا العزل مؤقت: بمجرد كتابة deactivate، تعود المتغيرات إلى حالتها الأصلية ويعود الأمر python إلى مفسر النظام. البيئة لا تغير النظام؛ إنها تغير فقط الطريقة التي يرى بها النظام الأوامر والحزم أثناء الجلسة الحالية.

3. Pinning Dependencies

3. تجميد الاعتماديات (Pinning Dependencies)

Creating the virtual environment is the first step. The second step is to record the dependencies the workflow is using in this environment. This recording is what allows you and others to rebuild the same environment at any time. The most common tool for recording dependencies in Python is pip freeze.

إنشاء البيئة الافتراضية هو الخطوة الأولى. الخطوة الثانية هي تسجيل الاعتماديات التي تستخدمها في هذه البيئة. هذا التسجيل هو ما يسمح لك وللآخرين بإعادة بناء البيئة نفسها في أي وقت. الأداة الأكثر شيوعاً لتسجيل الاعتماديات في Python هي pip freeze.

The pip freeze command prints a list of all packages installed in the current environment with their exact version numbers. By redirecting this output to a text file, usually named requirements.txt, you get a complete record of the project's dependencies. Then anyone else can use the command pip install -r requirements.txt to install the same versions in their own environment.

الأمر pip freeze يطبع قائمة بجميع الحزم المثبتة في البيئة الحالية مع أرقام إصداراتها الدقيقة. بتوجيه هذا الناتج إلى ملف نصي، عادةً باسم requirements.txt، تحصل على سجل كامل لاعتمادياتك. ثم يمكن لأي شخص آخر استخدام الأمر pip install -r requirements.txt لتثبيت نفس الإصدارات في بيئتهم الخاصة.

This file is not merely administrative; it is the digital fingerprint of the project's environment. If you lose the project's requirements.txt file, you will not know which versions you used, and you will not be able to rebuild the environment. And if you share the project's file with a colleague without this file, they will have to guess the appropriate versions, threatening reproducibility. Therefore, a simple rule: any bioinformatics project should contain a requirements.txt file (or a similar environment file) that is updated whenever you add or change a package.

هذا الملف ليس مجرد سجل إداري؛ إنه البصمة الرقمية لبيئتك. إذا فقدت ملف requirements.txt، فلن تعرف أي إصدارات استخدمتها، ولن تتمكن من إعادة بناء البيئة. وإذا شاركت ملفك مع زميل دون هذا الملف، فسيضطر إلى تخمين الإصدارات المناسبة، مما يهدد إعادة الإنتاج. لذلك، قاعدة بسيطة: أي مشروع معلوماتي حيوي يجب أن يحتوي على ملف requirements.txt (أو ملف بيئة مشابه) يُحدّث كلما أضفت أو غيّرت حزمة.

4. The Limits of Software-Only Reproducibility

4. حدود reproducibility البرمجية

It is important to understand that the virtual environment and requirements.txt only ensure the reproducibility of the software environment, not the complete environment. There are three other types of variables that may change the result even if the software environment is identical.

من المهم أن تفهم أن البيئة الافتراضية وملف requirements.txt يضمنان فقط إعادة إنتاج البيئة البرمجية، وليس البيئة الكاملة. هناك ثلاثة أنواع أخرى من المتغيرات التي قد تغير النتيجة حتى لو كانت البيئة البرمجية متطابقة:

First: Reference Data: Databases like GenBank or RefSeq are constantly changing. The same sequence may be stored under a single accession number, but the actual content may differ between versions. If you use a database without recording its date or version, you cannot reproduce the project's results.

أولاً: البيانات المرجعية (Reference Data) : قواعد البيانات مثل GenBank أو RefSeq تتغير باستمرار. قد يكون التسلسل نفسه مخزناً تحت رقم انضمام واحد، لكن المحتوى الفعلي قد يختلف بين الإصدارات. إذا استخدمت قاعدة بيانات دون تسجيل تاريخها أو إصدارها، فلا يمكن إعادة الإنتاج.

Second: Operating System and Architecture: Some packages behave differently on Windows compared to Linux, or on 64-bit compared to 32-bit systems. Even if the Python version is identical, results may differ due to underlying system differences.

ثانياً: نظام التشغيل والمعمارية (Operating System and Architecture) : بعض الحزم تتصرف بشكل مختلف على Windows مقارنة بـ Linux، أو على أنظمة 64-bit مقارنة بـ 32-bit. حتى لو كان إصدار Python متطابقاً، قد تختلف النتائج بسبب اختلافات النظام الأساسي.

Third: Parameters and Human Decisions: Reproducibility does not only mean re-running the code with the same data. It also means applying the same decisions: which threshold you chose, which normalization method you preferred, and how you handled missing values. If you do not document these decisions, others cannot understand why you reached the project's result.

ثالثاً: المعلمات والقرارات البشرية (Parameters and Human Decisions) : إعادة الإنتاج لا تعني فقط إعادة تشغيل الكود بنفس البيانات. تعني أيضاً تطبيق نفس القرارات: أي عتبة اخترت، وأي طريقة تطبيع فضلت، وكيف تعاملت مع القيم المفقودة. إذا لم توثق هذه القرارات، فلا يمكن للآخرين فهم سبب وصولك إلى نتيجتك.

For this reason, the virtual environment and requirements.txt are part of the solution, not the complete solution. They must be accompanied by documentation of reference data (with versions and dates), documentation of analysis decisions, and a record of the operating system. True reproducibility is an integrated project, not just package freezing.

لهذا السبب، البيئة الافتراضية و requirements.txt هما جزء من الحل، وليس الحل الكامل. يجب أن يرافقهما توثيق للبيانات المرجعية (مع الإصدارات والتواريخ) وتوثيق لقرارات التحليل، وسجل لنظام التشغيل. إعادة الإنتاج الحقيقية هي مشروع متكامل، وليس مجرد تجميد حزم.

5. Practical Application

5. التطبيق العملي

Let us apply what we have learned to a hypothetical project. For illustration, consider a case in which are working on sequence analysis using Biopython. Follow these steps.

لنطبق ما تعلمناه في مشروع وهمي. للتوضيح، يمكن تصور حالة يكون فيها تعمل على تحليل تسلسلات باستخدام Biopython. اتبع هذه الخطوات:

1. إنشاء بيئة جديدة:

Language / نوع الكود: **Bash** / **Shell**

```
python -m venv .venv
```

2. تفعيل البيئة:

Language / نوع الكود: **Bash** / **Shell**

```
source .venv/bin/activate  # Unix/macOS
.venv\Scripts\activate     # Windows
```

3. تثبيت الحزم التي تحتاجها:

Language / نوع الكود: **Bash** / **Shell**

```
pip install biopython numpy pandas
```

4. تجميد الاعتماديات في ملف:

Language / نوع الكود: **Bash** / **Shell**

```
pip freeze > requirements.txt
```

5. تضمين requirements.txt في مستودع مشروعك (مع الكود والبيانات).

6. عند مشاركة المشروع، يكتب زميلك:

Language / نوع الكود: **Bash** / **Shell**

```
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

وبذلك يحصل على نفس البيئة التي كنت تعمل عليها.

Chapter Summary

خلاصة الفصل

Virtual environments are the isolation tool that allows you to freeze the project's project's dependencies and prevent conflicts with other projects. The requirements.txt file is the digital fingerprint of these dependencies. But software

isolation alone does not guarantee complete reproducibility. Reference data, operating system, and human decisions must also be documented. A correct understanding of the limits of reproducibility is what distinguishes a professional researcher from a casual tool user.

البيئات الافتراضية هي أداة العزل التي تسمح لك بتجميد اعتماديات مشروعك ومنع تعارضها مع مشاريع أخرى. ملف requirements.txt هو البصمة الرقمية لهذه الاعتماديات. لكن عزل البرمجيات وحده لا يضمن إعادة الإنتاج الكاملة. يجب أيضاً توثيق البيانات المرجعية ونظام التشغيل والقرارات البشرية. الفهم الصحيح لحدود إعادة الإنتاج هو ما يميز الباحث المحترف عن المستخدم العادي للأدوات.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Virtual Environments and Packages" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "البيئات الافتراضية والحزم" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "the virtual environment fully guarantees the reproducibility of results." This is incorrect, because the virtual environment only isolates software. The correct statement is: "the virtual environment ensures consistency of package versions, but it does not ensure consistency of reference data, operating system, or human decisions." Complete reproducibility requires more than just package isolation.

مثال على التجاوز: القول إن "البيئة الافتراضية تضمن إعادة إنتاج النتائج بالكامل". هذا غير صحيح، لأن البيئة الافتراضية تعزل البرمجيات فقط. الصحيح هو: "البيئة الافتراضية تضمن اتساق إصدارات الحزم، لكنها لا تضمن اتساق البيانات المرجعية أو نظام التشغيل أو القرارات البشرية." إعادة الإنتاج الكاملة تحتاج إلى أكثر من مجرد عزل حزم.

Example: The presence of a precise requirements.txt file is a computational representation of the project’s software environment. But claiming that "this analysis proves a biological relationship" is not proven by the dependency file. The requirements.txt file tells us about the tools you used, not about the validity of the biological conclusion. Practical test: if you find an exciting result, Practical check: "Does this result depend on a specific library version, or is it a biological phenomenon that can be confirmed by independent evidenc?"e.

مثال: وجود ملف requirements.txt دقيق هو تمثيل حسابي لبيئتك البرمجية. لكن الادعاء بأن "هذا التحليل يثبت وجود علاقة بيولوجية" لا يثبته ملف الاعتماديات. ملف requirements.txt يخبرنا عن الأدوات التي

استخدمتها، وليس عن صحة الاستنتاج البيولوجي. اختبار عملي: إذا وجدت نتيجة مثيرة، فحص تطبيقي: "هل هذه النتيجة تعتمد على إصدار معين من مكتبة، أم أنها ظاهرة بيولوجية يمكن تأكيدها بأدلة مستقلة؟"ـ

· المدخلات: الأمر python -m venv .venv، وملف requirements.txt (إن وجد)ـ

مثال تطبيقي على البيئة الافتراضية:

· الافتراضات: أن الأمر python يشير إلى الإصدار الذي تريده، وأن البيئة لم تتعرض للتلف، وأن الحزم المطلوبة متوفرة في مستودعات PyPI بنفس الإصداراتـ

· المخرجات: بيئة معزولة تحتوي على مفسر Python ومجلد حزم مستقلـ

· القيود: لا تعزل البيئة الافتراضية نظام التشغيل، ولا البيانات المرجعية، ولا القرارات البشرية. هي أداة برمجية بحتةـ

Applied example of the virtual environment:

اختبار: قبل أن تبدأ أي مشروع، أنشئ بيئة جديدة، ثبّت الحزم، وولّد requirements.txt. إذا لم تستطع إعادة بناء البيئة من هذا الملف بعد عام، فاعلم أن هناك قيداً لم تأخذه في الحسبانـ

· Inputs: the command python -m venv .venv, and a requirements.txt file (if present).

· Assumptions: that the python command points to the version you want, that the environment has not been corrupted, and that the required packages are available on PyPI with the same versions.

· Outputs: an isolated environment containing a Python interpreter and an independent package folder.

· Limitations: the virtual environment does not isolate the operating system, nor reference data, nor human decisions. It is a purely software tool.

Test: before starting any project, create a new environment, install packages, and generate requirements.txt. If you cannot rebuild the environment from this file a year later, know that there is a limitation you did not account for.

عند إنشاء بيئة افتراضية، يجب تسجيل ما يلي:

· المصدر: نظام التشغيل وإصداره، وإصدار Python الذي استخدمته لإنشاء البيئةـ

· الإصدار: محتويات requirements.txt بالكامل (كل حزمة وإصدارها)ـ

· المعلمات: أي متغيرات بيئية أضفتها، وأي خيارات خاصة استخدمتها عند تثبيت الحزمـ

· التحويلات: أي خطوات إضافية قمت بها بعد إنشاء البيئة (مثل تثبيت حزم من خارج PyPI)ـ

When creating a virtual environment, record the following.

اختبار: فحص تطبيقي: "لو أعطيت هذا المجلد (مع requirements.txt) لزميل بعد عام، هل يستطيع إعادة بناء البيئة دون أن يسألني عن أي تفاصيل إضافية؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the operating system and its version, and the Python version you used to create the environment.

· Version: the complete contents of requirements.txt (every package and its version).

· Parameters: any environment variables you added, and any special options you used when installing packages.

· Transformations: any additional steps you took after creating the environment (e.g., installing packages from outside PyPI).

Test: Practical check: "If I gave this folder (with requirements.txt) to a colleague a year later, could they rebuild the environment without asking me for any additional details?" If the answer is no, the documentation is insufficient.

قبل أن تبدأ تحليلك الحقيقي، اختبر البيئة الافتراضية وآلية تجميد الاعتماديات على مشروع صغير وهمي. أنشئ بيئة، ثبّت حزمة واحدة (مثل numpy)، وولّد requirements.txt. ثم حاول إعادة بناء البيئة في مجلد آخر باستخدام هذا الملف. إذا نجحت، تكون واثقاً من أن آلية عملك صحيحة. إذا فشلت، فستعرف المشكلة في طريقة تجميدك أو في بيئتك الأساسية. هذه العادة توفر ساعات من البحث في المستقبل.

Before you start the project’s real analysis, test the virtual environment and dependency freezing mechanism on a small dummy project. Create an environment, install one package (e.g., numpy), and generate requirements.txt. Then try to rebuild the environment in another folder using this file. If you succeed, the workflow is confident the project’s workflow is correct. If you fail, you will know the problem is in the project’s freezing method or the project’s base environment. This habit saves hours of future debugging.

Chapter 6 — First Encounter with Biopython

الفصل 6 — أول لقاء مع Biopython

الفكرة المركزية: فلسفة المكتبة، الكائنات الحيوية، API، التوثيق.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

القيمة التعليمية في Biopython تظهر عندما يصبح الفرق بين النص الخام والكائن البيولوجي واضحاً. Seq يحتفظ بعمليات مثل التكامل العكسي والترجمة، لكن هذه العمليات تبقى حسابية. الانتقال من نتيجة مثل بروتين مترجم إلى قول إن بروتيناً وظيفياً موجود فعلاً يحتاج إلى أدلة مستقلة وسياق بيولوجي.

Biopython becomes especially instructive when the distinction between raw text and a biological object is made explicit. Seq provides operations such as reverse complementation and translation, but these operations remain computational. Moving from a translated protein sequence to a claim that a functional protein actually exists requires independent evidence and biological context.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. ما هو Biopython؟

1. What Is Biopython؟

Biopython is an open-source Python library specifically designed for handling biological data. It provides tools for reading and writing various file formats (such as FASTA and GenBank), representing sequences and features, performing basic biological operations like reverse complementation and translation, and connecting to public databases. But more importantly, Biopython is not just a collection of functions; it is a philosophy: representing biological entities as software objects that carry their meaning and context.

Biopython هي مكتبة Python مفتوحة المصدر صُنعت خصيصاً للتعامل مع البيانات البيولوجية. توفر أدوات لقراءة وكتابة صيغ الملفات المختلفة (مثل FASTA و GenBank)، وتمثيل التسلسلات والميزات، وإجراء العمليات البيولوجية الأساسية مثل التكامل العكسي والترجمة، والاتصال بقواعد البيانات العامة. لكن الأهم من ذلك، Biopython ليست مجرد مجموعة دوال؛ إنها فلسفة: تمثيل الكيانات البيولوجية ككائنات برمجية تحمل معناها وسياقها.

Imagine dealing with a DNA sequence as a plain string. the workflow can reverse it, but you cannot compute its biological complement or know its reading frame. Biopython changes this entirely: it provides you with a Seq object that knows it represents a biological sequence, knows the rules of complementarity, knows orientation, and knows how to translate to protein. This shift from "text" to "biological object" is the difference between general programming and bioinformatics.

للتوضيح، يمكن تصور حالة يكون فيها تتعامل مع تسلسل DNA كمجرد سلسلة نصية. يمكن عكسها، لكن لا يمكن حساب مكملها البيولوجي أو معرفة إطارها القرائي. Biopython تغيّر هذا تماماً: فهي تقدم لك كائن Seq الذي يعرف أنه يمثل تسلسلاً بيولوجياً، ويعرف قواعد التكامل، ويعرف الاتجاهية، ويعرف كيف يترجم إلى بروتين. هذا التحول من "نص" إلى "كائن حيوي" هو الفرق بين البرمجة العامة والمعلوماتية الحيوية.

2. The Fundamental Biological Objects

2. الكائنات الحيوية الأساسية

Biopython relies on three fundamental objects, which you saw briefly in the preliminary chapter, but now we understand them in their deep programming context.

تعتمد Biopython على ثلاثة كائنات أساسية، رأيتها بإيجاز في الفصل التمهيدي، لكننا الآن نفهمها في سياقها البرمجي العميق:

First: Seq — This is the fundamental object. It represents the sequence itself (a string of letters), but it provides operations with biological meaning: complement(), reverse_complement(), translate(), and others. It is not just text; it is an intelligent representation of the molecule.

أولاً: Seq — هذا هو الكائن الأساسي. يمثل التسلسل نفسه (سلسلة من الحروف)، لكنه يقدم عمليات ذات معنى بيولوجي: translate، reverse_complement()، complement()، وغيرها. هو ليس مجرد نص؛ هو تمثيل ذكي للجزيء.

Second: SeqRecord — This object wraps the sequence with an identity (id), description (description), and metadata (annotations). It is the complete record: the sequence with its birth certificate. In practice, when you read a FASTA file using Biopython, you get a SeqRecord object, not just a Seq. This ensures that identity and description travel with the sequence wherever they go.

ثانياً: SeqRecord — هذا الكائن يغلف التسلسل بهوية (id)، ووصف (description)، وبيانات وصفية (annotations). هو السجل الكامل: التسلسل مع شهادة ميلاده. في الممارسة العملية، عند قراءة ملف FASTA باستخدام Biopython، تحصل على كائن SeqRecord، وليس مجرد Seq. هذا يضمن أن الهوية والوصف يسافران مع التسلسل أينما ذهب.

Third: SeqFeature — This object represents a localized feature on the sequence: start and end positions, strand, and type (such as gene, exon, coding region). In the Biopython context, SeqFeature is how you represent additional information about the sequence, such as gene locations or regulatory regions.

ثالثاً: SeqFeature — هذا الكائن يمثل ميزة موضعية على التسلسل: موقع بداية ونهاية، اتجاه، ونوع (مثل جين، إكسون، منطقة مشفرة). في سياق Biopython، SeqFeature هي الطريقة التي تمثل بها المعلومات الإضافية حول التسلسل، مثل مواقع الجينات أو المناطق التنظيمية.

These three objects work together hierarchically: Seq is the fundamental building block, SeqRecord is the envelope that adds identity, and SeqFeature is the marker that identifies important locations. Understanding this hierarchical structure is the first step to understanding Biopython.

هذه الكائنات الثلاثة تعمل معاً بشكل هرمي: Seq هو اللبنة الأساسية، SeqRecord هو المظروف الذي يضيف الهوية، و SeqFeature هو العلامة التي تحدد المواقع المهمة. فهم هذا الهيكل الهرمي هو الخطوة الأولى لفهم Biopython.

3. API and Documentation

3. API والتوثيق

Biopython is a vast library containing hundreds of functions and modules. It is neither possible nor advisable to memorize everything. The real skill is not knowing every function, but knowing where to look when the analysis requires it. The official Biopython documentation is the primary reference. When a Biopython version differs or an API changes, the correct reference is the official documentation and the version the workflow is using, not memory or an old example.

Biopython هي مكتبة ضخمة تضم مئات الدوال والوحدات. لا يمكن، ولا ينبغي، حفظ كل شيء. المهارة الحقيقية ليست في معرفة كل دالة، بل في معرفة أين تبحث عنها عندما تحتاجها. وثائق Biopython الرسمية هي المرجع الأساسي. عندما تختلف نسخة Biopython أو تتغير واجهة برمجية (API)، المرجع الصحيح هو التوثيق الرسمي والإصدار المستخدم، وليس الذاكرة أو مثالاً قديماً.

The educational rule here is simple and consistent: read the documentation before you expect behavior. Before you use a new function, read its page in the documentation. Learn its parameters, defaults, return values, and possible error conditions. This habit is not just professionalism; it is the only way to deal with any constantly evolving library. Biopython changes, improves, and fixes, and yesterday's version may not be today's.

القاعدة التعليمية هنا بسيطة وثابتة: اقرأ الوثيقة قبل أن تتوقع السلوك. قبل أن تستخدم دالة جديدة، اقرأ صفحتها في التوثيق. تعرّف على معاملاتها، وقيمها الافتراضية، وقيمها المرتجعة، وحالات الخطأ المحتملة. هذه العادة ليست مجرد

احترافية؛ إنها الطريقة الوحيدة للتعامل مع أي مكتبة تتطور باستمرار. Biopython تتغير، وتتحسن، وتُصلح، وإصدار الأمس قد لا يكون كإصدار اليوم.

4. The Difference Between API and Biological Inference

4. الفرق بين API والاستدلال البيولوجي

A common mistake is for a beginner to assume that Biopython's ability to perform a computational operation (such as translation to protein) means that the output represents a biological truth. This is not correct. Biopython performs specific computational operations: it reads a sequence, applies translation rules according to a specific genetic code table, and produces a string of amino acids. This is what the library does. The why this output has biological meaning, or whether it represents a real functional protein in the cell, is a biological inference that requires additional knowledge, context, and independent experiments.

من الأخطاء الشائعة أن يعتقد المبتدئ أن قدرة Biopython على تنفيذ عملية حسابية (مثل الترجمة إلى بروتين) تعني أن الناتج يمثل حقيقة بيولوجية. هذا ليس صحيحاً. Biopython تنفذ عمليات حسابية محددة: تقرأ تسلسلاً، وتطبّق قواعد الترجمة وفق جدول شفرة وراثية معين، وتنتج سلسلة من الأحماض الأمينية. هذا هو "ما" تفعله المكتبة. أما "لماذا" هذا الناتج ذو معنى بيولوجي، أو ما إذا كان يمثل بروتيناً وظيفياً حقيقياً في الخلية، فهذا استدلال بيولوجي يتطلب معرفة إضافية وسياقاً وتجارب مستقلة.

After the fundamental objects comes the file handling tool: SeqIO. This module allows you to read and write multiple sequence file formats, including FASTA and GenBank. When reading a file using SeqIO.read(), you get a single SeqRecord object. When reading a file containing multiple sequences, you use SeqIO.parse(), which returns an iterator of SeqRecord objects. These tools are the practical gateway to working with real data.

بعد الكائنات الأساسية، تأتي أداة التعامل مع الملفات: SeqIO. هذه الوحدة تسمح لك بقراءة وكتابة صيغ متعددة من ملفات التسلسل، بما في ذلك FASTA و GenBank. عند قراءة ملف باستخدام SeqIO.read()، تحصل على كائن SeqRecord واحد. أما عند قراءة ملف يحتوي على عدة تسلسلات، فتستخدم SeqIO.parse() التي تعيد مكرراً (iterator) من SeqRecord. هذه الأدوات هي البوابة العملية للتعامل مع البيانات الحقيقية.

This chapter distinguishes between the API (what the library can do programmatically) and biological inference (what we infer about the living organism from these results). Biopython is a tool for generating computational evidence, not a means of proving biological facts. Distinguishing between the two is the essence of the scientific method in bioinformatics.

يميز هذا الفصل بين API (ما يمكن للمكتبة أن تفعله برمجياً) وبين الاستدلال البيولوجي (ما نستنتجه عن الكائن الحي من هذه النتائج). Biopython هي أداة لتوليد الأدلة الحسابية، وليست وسيلة لإثبات الحقائق البيولوجية.

الفصل بين الاثنين هو جوهر المنهج العلمي في المعلوماتية الحيوية.

5. Practical Application

5. التطبيق العملي

Let us apply this in a simple exercise. In the project's virtual environment, open Python and type:

دعنا نطبق هذا في تمرين بسيط. في بيئتك الافتراضية، افتح Python واكتب:

```
from Bio.Seq import Seq
# اصنع تسلسلاً اصطناعياً
dna = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC")
# العمليات الحسابية
print("المتمم:", dna.complement())
print("المتمم العكسي:", dna.reverse_complement())
print("الترجمة:", dna.translate())
```

Notice that translate() translated the sequence to protein according to the first reading frame. This is computationally correct. But does this protein actually exist in nature? Does it perform a function? Biopython does not tell you that. the analysis requires context, databases, comparison, and independent evidence. This is the difference between a computational operation and biological inference.

لاحظ أن translate() قام بترجمة التسلسل إلى بروتين وفق الإطار القرائي الأول. هذا صحيح حسابياً. لكن هل هذا البروتين موجود فعلاً في الطبيعة؟ هل يؤدي وظيفة؟ Biopython لا تخبرك بذلك. تحتاج إلى سياق، وقاعدة بيانات، ومقارنة، وأدلة مستقلة. هذا هو الفرق بين العملية الحسابية والاستدلال البيولوجي.

Chapter Summary

خلاصة الفصل

Biopython is not just a collection of functions; it is a philosophy of representing biological entities as intelligent software objects. The three fundamental objects — Seq, SeqRecord, and SeqFeature — form a hierarchical structure that allows rich data representation. But true power comes from understanding the limits: Biopython performs computational operations, and biological inference is the researcher's responsibility. Official documentation is the reference, and distinguishing between API and biology is the essence of critical thinking. The next

chapter will build on these foundations and introduce you to basic Python data types and how to use them in a biological context.

Biopython هي ليست مجرد مجموعة دوال؛ إنها فلسفة تمثيل الكيانات البيولوجية ككائنات برمجية ذكية. الكائنات الثلاثة الأساسية — SeqRecord ،Seq، و SeqFeature — تشكل هيكلاً هرمياً يسمح بتمثيل غني للبيانات. لكن القوة الحقيقية تأتي من فهم الحدود: Biopython تنفذ عمليات حسابية، والاستدلال البيولوجي هو مسؤولية الباحث. الوثائق الرسمية هي المرجع، والتمييز بين API والبيولوجيا هو جوهر التفكير النقدي. الفصل القادم سيبني على هذه الأسس، ويقدم يمكن أنواع البيانات الأساسية في Python وكيفية استخدامها في السياق البيولوجي.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "First Encounter with Biopython" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "أول لقاء مع Biopython" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "Biopython can prove the existence of a functional protein in the cell." This is incorrect; Biopython performs a computational translation, but it does not prove function. The correct statement is: "Biopython provides tools for representing sequences and performing computational operations on them, but interpreting the results biologically requires context and independent evidence." A simple test: after you translate a sequence using Biopython, Practical check: "Is this protein known in databases? Is there experimental evidence for its functio?"n.

مثال على التجاوز: القول إن "Biopython يمكن إثبات وجود بروتين وظيفي في الخلية". هذا غير صحيح؛ Biopython تنفذ ترجمة حسابية، لكنها لا تثبت الوظيفة. الصحيح هو: "Biopython توفر أدوات لتمثيل التسلسلات وإجراء عمليات حسابية عليها، لكن تفسير النتائج بيولوجياً يحتاج إلى سياق وأدلة مستقلة." اختبار بسيط: بعد أن تترجم تسلسلاً باستخدام Biopython، فحص تطبيقي: "هل هذا البروتين معروف في قواعد البيانات؟ هل هناك أدلة تجريبية على وظيفته؟"ـ

Example: Computing the reverse complement using dna.reverse_complement() is a purely computational representation. Claiming that "this complementary sequence actually exists in the cell" is a biological claim not justified by the computational operation alone. The reverse complement is a mathematical operation on letters, not a biological discovery. Practical test: before converting any Biopython result into a biological claim, verify that there is independent evidence (such as a database or published study) supporting that claim.

مثال: حساب المتمم العكسي باستخدام dna.reverse_complement() هو تمثيل حسابي بحت. الادعاء بأن "هذا التسلسل المتمم موجود فعلاً في الخلية" هو ادعاء بيولوجي غير مبرر بالعملية الحسابية وحدها. المتمم العكسي هو عملية رياضية على الحروف، وليس اكتشافاً بيولوجياً. اختبار عملي: قبل أن تحوّل أي نتيجة لـ Biopython إلى ادعاء بيولوجي، تأكد من وجود دليل مستقل (مثل قاعدة بيانات أو دراسة منشورة) يدعم هذا الادعاء.

مثال تطبيقي على دالة translate() في Biopython:

· المدخلات: كائن Seq يحتوي على تسلسل DNA، وجدول شفرة وراثية (افتراضياً الجدول القياسي).

· الافتراضات: أن التسلسل يبدأ بإطار قراءة صحيح، وأنه لا يحتوي على أكواد توقف غير متوقعة، وأن جدول الشفرة الوراثية مناسب للكائن الحي المدروس.

· المخرجات: كائن Seq يحتوي على سلسلة الأحماض الأمينية المترجمة.

· القيود: translate() لا تتحقق من صحة الإطار القرائي، ولا تكتشف أخطاء التسلسل، ولا تأخذ في الاعتبار التعديلات اللاحقة للترجمة (post-translational modifications). هي أداة حسابية بحتة.

Applied example of the translate() function in Biopython.

اختبار: قبل أن تستخدم translate()، تحقق من طول التسلسل وتأكد من عدم وجود أكواد توقف مبكرة. إذا ظهرت أكواد توقف غير متوقعة، فقد يكون التسلسل غير صحيح أو الإطار القرائي خاطئاً.

· Inputs: a Seq object containing a DNA sequence, and a genetic code table (default is the standard table).

· Assumptions: that the sequence starts in the correct reading frame, that it contains no unexpected stop codons, and that the genetic code table is appropriate for the organism under study.

· Outputs: a Seq object containing the translated amino acid sequence.

· Limitations: translate() does not verify the reading frame, nor detect sequencing errors, nor account for post-translational modifications. It is a purely computational tool.

Test: before using translate(), check the sequence length and ensure there are no premature stop codons. If unexpected stop codons appear, the sequence may be incorrect or the reading frame may be wrong.

عند استخدام Biopython، يجب تسجيل ما يلي:

· المصدر: التسلسل المستخدم (رقم الانضمام، الكائن الحي، النسيج).

· الإصدار: إصدار Biopython (مثل 1.81)، وإصدار Python، وإصدار جدول الشفرة الوراثية المستخدم.

· المعلمات: جدول الشفرة الوراثية (إذا كان مختلفاً عن الافتراضي)، وأي معاملات أخرى استخدمتها في الدوال.

· التحويلات: أي خطوات قمت بها قبل الترجمة (مثل قص التسلسل، أو تغيير الإطار القرائي).

When using Biopython, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة نفس الترجمة باستخدام نفس الإصدارات ونفس المعلمات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the sequence used (accession number, organism, tissue).

· Version: Biopython version (e.g., 1.81), Python version, and the genetic code table version used.

· Parameters: the genetic code table (if different from default), and any other parameters you used in functions.

· Transformations: any steps you performed before translation (e.g., trimming the sequence, or changing the reading frame).

Test: Practical check: "If I gave this record to a colleague a year later, could they reproduce the same translation using the same versions and parameters?" If the answer is no, the documentation is insufficient.

قبل أن تستخدم Biopython على تسلسل حقيقي من مريض أو من قاعدة بيانات، اختبر دوال Biopython على تسلسل اصطناعي صغير معروف النتيجة. استخدم التسلسل "ATGC" وتأكد أن reverse_complement() يعطي "GCAT". استخدم التسلسل "ATGGCC" (الذي يترجم إلى "MA") وتأكد أن translate() يعطي النتيجة المتوقعة. إذا نجحت هذه الاختبارات، فأنت واثق من أن المكتبة تعمل كما هو متوقع، وأي نتيجة تحصل عليها من بيانات حقيقية تعكس خصائص البيانات وليس خطأ في استخدامك للمكتبة.

Before using Biopython on a real sequence from a patient or a database, test Biopython's functions on a small synthetic sequence with a known outcome. Use the sequence "ATGC" and verify that reverse_complement() returns "GCAT". Use the sequence "ATGGCC" (which translates to "MA") and verify that translate() gives the expected result. If these tests succeed, the workflow is confident that the library works as expected, and any result you get from real data reflects the properties of the data, not an error in the project’s use of the library.

Part 3 — Python as the Language Foundation for Computational Biology

الباب 3 — Python كأساس لغوي للبيولوجيا الحاسوبية

الفكرة المركزية: strings، lists، dictionaries، sets، الأنواع والقيود والتحقق.

Chapter 7 — Data Types

الفصل 7 — أنواع البيانات

شرح أعمق للمفهوم

Deeper Conceptual Explanation

اختيار نوع البيانات ليس قراراً شكلياً في Python؛ فهو يحدد العمليات التي تصبح طبيعية وآمنة، ويحدد أنواع الأخطاء التي يمكن اكتشافها مبكراً. في المعلوماتية الحيوية، يجب أن يبقى الفرق واضحاً بين نص يمثل تسلسلاً، وقائمة تمثل مجموعة قيم، وقاموس يربط معرفات ببيانات وصفية.

Choosing a data type is not merely a Python style decision; it determines which operations are natural and which errors can be caught early. In bioinformatics, the distinction between text representing a sequence, a list representing values, and a dictionary linking identifiers to metadata must remain explicit.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Strings

1. السلاسل النصية (Strings)

A string is the simplest and most widely used type in bioinformatics. A DNA sequence, in its simplest form, is a string of the letters A, T, C, G. In Python, text is written between single or double quotation marks: "ATGC". Strings allow you to store the sequence, display it, search it, and compare it with other sequences. But they carry no additional biological meaning; they are merely letters. The real power comes when we use strings with logical operations, such as searching for a specific pattern, calculating length, or extracting a substring.

السلسلة النصية هي النوع الأبسط والأكثر استخداماً في المعلوماتية الحيوية. التسلسل DNA، في أبسط صوره، هو سلسلة نصية من الحروف A، T، C، G. في Python، النص يُكتب بين علامتي تنصيص مفردة أو مزدوجة: "ATGC". السلاسل النصية تسمح لك بتخزين التسلسل، وعرضه، والبحث فيه، ومقارنته بتسلسلات أخرى. لكنها لا

تحمل أي معنى بيولوجي إضافي؛ هي مجرد حروف. القوة الحقيقية تأتي عندما نستخدم السلاسل النصية مع العمليات المنطقية، مثل البحث عن نمط معين، أو حساب الطول، أو استخراج جزء من التسلسل.

```
# مثال: التعامل مع التسلسل كنص عادي
sequence = "ATGCGTACGTAGCTAGCTAGCTAGCTAGC"
print(len(sequence))      # طول التسلسل: 30
print(sequence[0:6])      # أول 6 قواعد: ATGCGT
print("TAG" in sequence)  # هل يحتوي على TAG؟ True
```

2. Lists

2. القوائم (Lists)

Lists are a data type that can store a collection of elements in a specific order. In bioinformatics, lists are useful for organizing samples, aggregating multiple results, or storing a list of identifiers. A list can contain any type of data: strings, numbers, or even other objects. Lists are mutable: the workflow can add elements, delete them, or change their order. This makes them highly flexible, but requires extra attention to maintain relationships between data. For example, if the workflow has a list of identifiers and a list of sequences, you must ensure that their order matches; otherwise, you lose the relationship between identifier and sequence.

القوائم هي نوع بيانات يمكنه تخزين مجموعة من العناصر بترتيب محدد. في المعلوماتية الحيوية، القوائم مفيدة لتنظيم العينات، أو تجميع نتائج متعددة، أو تخزين قائمة من المعرفات. يمكن للقائمة أن تحتوي على أي نوع من البيانات: نصوص، أرقام، أو حتى كائنات أخرى. القوائم قابلة للتعديل: يمكن إضافة عناصر، حذفها، أو تغيير ترتيبها. هذا يجعلها مرنة للغاية، لكنها تتطلب انتباهاً إضافياً للحفاظ على العلاقات بين البيانات. على سبيل المثال، إذا كانت يوجد في المثال قائمة من المعرفات وقائمة من التسلسلات، يجب أن تتأكد من أن ترتيبهما متطابق، وإلا فقدت العلاقة بين المعرف والتسلسل.

```
# قائمة من المعرفات
sample_ids = ["S1", "S2", "S3", "S4"]
sequences = ["ATGC", "CGTA", "GCAT", "TACG"]
# العلاقة بين المعرف والتسلسل تعتمد على الترتيب
print(sample_ids[0], sequences[0])  # S1 ATGC
```

3. Dictionaries

Dictionaries are a data type that associates each key with a value. This is the ideal type for metadata, where each sample has a unique identifier (the key) and a set of associated data (the value). Instead of relying on list order, dictionaries allow you to access data directly using the identifier. This protects the relationship between the sample and its data, even if the order of elements changes. In bioinformatics,

dictionaries are widely used to store sample metadata, analysis results, or any data that needs to associate an identifier with a value.

3. القواميس (Dictionaries)

القواميس هي نوع بيانات يربط كل مفتاح (key) بقيمة (value). هذا هو النوع المثالي للبيانات الوصفية، حيث يكون لكل عينة معرف فريد (المفتاح) ومجموعة من البيانات المرتبطة به (القيمة). بدلاً من الاعتماد على ترتيب القوائم، تسمح لك القواميس بالوصول إلى البيانات مباشرة باستخدام المعرف. هذا يحمي العلاقة بين العينة وبياناتها، حتى لو تغير ترتيب العناصر. في المعلوماتية الحيوية، القواميس تستخدم بشكل واسع لتخزين البيانات الوصفية للعينات، أو نتائج التحليل، أو أي بيانات تحتاج إلى ربط معرف بقيمة.

```
# قاموس يربط معرف العينة بالبيانات الوصفية
sample_metadata = {
    "S1": {"tissue": "liver", "age": 45, "sex": "M"},
    "S2": {"tissue": "kidney", "age": 32, "sex": "F"},
    "S3": {"tissue": "liver", "age": 28, "sex": "M"}
}
# الوصول إلى بيانات العينة S1
print(sample_metadata["S1"])  # {'tissue': 'liver', 'age': 45, 'sex': 'M'}
```

4. Sets

Sets are a data type that stores unique (non-duplicated) elements. This is very useful in bioinformatics when you want to ensure that identifiers are unique, or when you want to extract unique values from a large dataset. For example, if the workflow has a list of identifiers and you want to know which ones are unique, the workflow can convert it to a set. Sets do not maintain order, but they are very fast for searching and checking for membership.

4. المجموعات (Sets)

المجموعات هي نوع بيانات يخزن عناصر فريدة (غير مكررة). هذا مفيد جداً في المعلوماتية الحيوية عندما تريد التأكد من أن المعرفات فريدة، أو عندما تريد استخراج القيم الفريدة من مجموعة بيانات كبيرة. على سبيل المثال، إذا كانت يوجد في المثال قائمة من المعرفات وتريد معرفة أي منها فريد، يمكن تحويلها إلى مجموعة. المجموعات لا تحافظ على الترتيب، لكنها سريعة جداً في البحث عن العناصر والتحقق من وجودها.

```
# قائمة قد تحتوي على معرفات مكررة
sample_ids = ["S1", "S2", "S3", "S1", "S4", "S2"]
unique_ids = set(sample_ids)
print(unique_ids)  # {'S1', 'S2', 'S3', 'S4'}
# التحقق من وجود معرف فريد
if "S1" in unique_ids:
    print("S1 موجودة في المجموعة")
```

5. Choosing the Appropriate Structure

Choosing the appropriate structure is not just a personal preference; it is a scientific decision that affects the validity of the project's analysis. If you use a string to store multiple identifiers, you lose the ability to access them individually. If you use a list to store metadata, you depend on order, which can break if the data changes. If you use a dictionary, you preserve the relationship between identifier and data. If you use a set, the workflow can verify uniqueness. The right structure makes constraints explicit: if an identifier must be unique, a set tests that. If the relationship between sample and data is important, a dictionary preserves it. The wrong structure may hide errors and lead to incorrect conclusions.

5. اختيار البنية المناسبة

6. Practical Application: Comparing Structures

اختيار البنية المناسبة ليس مجرد تفضيل شخصي؛ هو قرار علمي يؤثر على صحة تحليلك. إذا استخدمت سلسلة نصية لتخزين معرفات متعددة، فستفقد القدرة على الوصول الفردي. إذا استخدمت قائمة لتخزين بيانات وصفية، فستعتمد على الترتيب، وهذا قد ينكسر إذا تغيرت البيانات. إذا استخدمت قاموساً، فستحافظ على العلاقة بين المعرف والبيانات. إذا استخدمت مجموعة، فيمكن التحقق من التفرد. البنية الصحيحة تجعل القيود ظاهرة: إذا كان المعرف يجب أن يكون فريداً، فالمجموعة تختبر ذلك. إذا كانت علاقة العينة بالبيانات مهمة، فالقالوس يحفظها. البنية الخاطئة قد تخفي الأخطاء وتؤدي إلى استنتاجات خاطئة.

Let us apply this to a real-world example. For illustration, consider a case in which have data from a gene expression experiment. the workflow has sample identifiers, patient ages, tissues, and expression results. How do you store this dat? a.

6. التطبيق العملي: مقارنة بين البنى

Wrong way (separate lists): Four separate lists, depending on order. If any list changes order, relationships are lost.

دعنا نطبق هذا على مثال واقعي. تخيل أن يوجد في المثال بيانات من تجربة تعبير جيني. يوجد في المثال معرفات العينات، وأعمار المرضى، وأنسجتهم، ونتائج التعبير. كيف تخزن هذه البيانات؟

الطريقة الخاطئة (قوائم منفصلة): أربع قوائم منفصلة، تعتمد على الترتيب. إذا تغير ترتيب أي قائمة، فستفقد العلاقات.

ids = ["S1", "S2", "S3"]
ages = [45, 32, 28]
tissues = ["liver", "kidney", "liver"]
expression = [2.5, 1.8, 3.1]
# إذا تغير ترتيب ids، فلا نعرف أي عمر ينتمي لأي عينة

Right way (dictionary): Key is the sample identifier, value is another dictionary containing all data.

الطريقة الصحيحة (قاموس): مفتاح هو معرف العينة، والقيمة هي قاموس آخر يحوي كل البيانات.

```
samples = {
    "S1": {"age": 45, "tissue": "liver", "expression": 2.5},
    "S2": {"age": 32, "tissue": "kidney", "expression": 1.8},
    "S3": {"age": 28, "tissue": "liver", "expression": 3.1}
}
# الآن العلاقات محفوظة، بغض النظر عن الترتيب
print(samples["S2"]["expression"])  # 1.8
```

7. Data Validation

The right structure makes validation possible. If an identifier must be unique, a set tests that. If age must be a positive number, the workflow can check that before storing the data. Data validation is not an extra step; it is part of the representation itself. Choosing the appropriate structure allows you to write simple validation functions that detect errors before they reach the analysis. This is the essence of defensive programming: assume the data may be wrong, and test the project's assumptions.

7. التحقق من صحة البيانات (Validation)

البنية الصحيحة تجعل التحقق ممكناً. إذا كان المعرف يجب أن يكون فريداً، فالمجموعة تختبر ذلك. إذا كان يجب أن يكون العمر رقماً موجباً، فيمكن التحقق من ذلك قبل تخزين البيانات. التحقق من صحة البيانات ليس خطوة إضافية؛ إنه جزء من التمثيل نفسه. اختيار البنية المناسبة يسمح لك بكتابة دوال تحقق بسيطة تكتشف الأخطاء قبل أن تصل إلى التحليل. هذا هو جوهر البرمجة الدفاعية: افترض أن البيانات قد تكون خاطئة، واختبر افتراضاتك.

Chapter Summary

خلاصة الفصل

Python data types are the fundamental tools upon which all bioinformatics structures are built. Strings store sequences, lists organize samples, dictionaries preserve relationships, and sets ensure uniqueness. Choosing the appropriate structure is not a purely programming decision, but a scientific one that affects the validity of the project's analysis. The right structure makes constraints explicit and validation possible. The wrong

structure hides errors and leads to incorrect conclusions. The next chapter will build on this foundation to introduce you to functions, modules, and error handling, as tools for organizing and validating code.

أنواع البيانات في Python هي الأدوات الأساسية التي تبنى عليها كل بنى المعلوماتية الحيوية. السلاسل النصية تخزن التسلسلات، والقوائم تنظم العينات، والقواميس تحافظ على العلاقات، والمجموعات تضمن التفرد. اختيار البنية المناسبة ليس قراراً برمجياً بحتاً، بل هو قرار علمي يؤثر على صحة تحليلك. البنية الصحيحة تجعل القيود ظاهرة والتحقق ممكناً. البنية الخاطئة تخفي الأخطاء وتؤدي إلى استنتاجات خاطئة. الفصل القادم سيبني على هذا الأساس ليقدم لك الدوال والوحدات وإدارة الأخطاء، كأدوات لتنظيم الكود والتحقق من صحته.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Data Types" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "أنواع البيانات" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "choosing the right data type guarantees the correctness of biological results." This is incorrect; the right data type ensures sound representation, but the correctness of biological results depends on data quality and analysis. The correct statement is: "choosing the appropriate data type makes representation accurate and validation possible, but it does not compensate for poor data or flawed analysis." A simple test: Practical check: "Does my structure clearly store the relationships between data, or am I relying on order or guesswor?"k.

مثال على التجاوز: القول إن "اختيار نوع البيانات الصحيح يضمن صحة النتائج البيولوجية". هذا غير صحيح؛ نوع البيانات الصحيح يضمن تمثيلاً سليماً، لكن صحة النتائج البيولوجية تعتمد على جودة البيانات والتحليل. الصحيح هو: "اختيار نوع البيانات المناسب يجعل التمثيل دقيقاً والتحقق ممكناً، لكنه لا يعوض عن بيانات رديئة أو تحليل خاطئ." اختبار بسيط: فحص تطبيقي: "هل بنيتي تخزن العلاقات بين البيانات بشكل واضح، أم أنني أعتمد على الترتيب أو التخمين؟".

Example: Storing sample identifiers in a dictionary is a computational representation. Claiming that "these samples represent a homogeneous group of patients" is a biological claim not proven by the data type. The dictionary tells us about the relationship between identifier and data, not about the biological significance of the samples. Practical test: before interpreting any dataset, ensure that its computational representation does not hide important biological differences.

مثال: تخزين معرفات العينات في قاموس هو تمثيل حسابي. الادعاء بأن "هذه العينات تمثل مجموعة متجانسة من المرضى" هو ادعاء بيولوجي لا يثبته نوع البيانات. القاموس يخبرنا عن العلاقة بين المعرف والبيانات، وليس عن دلالة العينات البيولوجية. اختبار عملي: قبل أن تفسر أي مجموعة بيانات، تأكد من أن تمثيلها الحسابي لا يخفي اختلافات بيولوجية مهمة.

مثال تطبيقي على تخزين البيانات الوصفية في قاموس:

· المدخلات: معرفات فريدة، وبيانات وصفية (عمر، نسيج، تعبير).

· الافتراضات: أن المعرفات فريدة وغير مكررة، وأن البيانات الوصفية مكتملة وصحيحة.

· المخرجات: قاموس يربط كل معرف ببياناته الوصفية.

· القيود: القاموس لا يتحقق من صحة البيانات (مثل أن العمر رقم موجب)، ولا يفرض وجود حقول معينة.

Applied example of storing metadata in a dictionary.

اختبار: قبل تخزين البيانات، تحقق من أن جميع المعرفات فريدة، وأن جميع الحقول المطلوبة موجودة.

· Inputs: unique identifiers, and metadata (age, tissue, expression).

· Assumptions: that identifiers are unique and not duplicated, and that metadata is complete and correct.

· Outputs: a dictionary linking each identifier to its metadata.

· Limitations: the dictionary does not validate the data (e.g., that age is a positive number), nor does it enforce the presence of specific fields.

Test: before storing data, verify that all identifiers are unique and that all required fields are present.

عند استخدام هياكل البيانات، يجب تسجيل:

· المصدر: كيف جُمعت البيانات، ومن أي مصدر.

· الإصدار: إصدار Python المستخدم، وأي مكتبات استخدمتها لمعالجة البيانات.

· المعلمات: أي اختيارات قمت بها لتنظيم البيانات (مثل ترتيب القوائم، أو مفاتيح القواميس).

· التحويلات: أي تحويلات طبقت على البيانات الخام قبل تخزينها.

When using data structures, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا الهيكل لزميل بعد عام، هل يستطيع فهم كيف نُظمت البيانات ولماذا؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: how the data was collected, and from what source.

· Version: Python version used, and any libraries used to process the data.

· Parameters: any choices you made in organizing the data (e.g., list order, dictionary keys).

· Transformations: any transformations applied to the raw data before storage.

Test: Practical check: "If I gave this structure to a colleague a year later, could they understand how the data was organized and why?" If the answer is no, the documentation is insufficient.

قبل أن تخزن بيانات حقيقية في هياكل البيانات، اختبر بنيتك على بيانات اصطناعية صغيرة. أنشئ قاموساً يحتوي على 3-4 عينات وهمية، وتحقق من أنه يعمل كما هو متوقع. ثم حاول إضافة بيانات جديدة، وحذف بيانات، والوصول إلى البيانات عبر المعرفات. إذا نجحت هذه الاختبارات، فأنت واثق من أن بنيتك صحيحة، وأن أي مشكلة في البيانات الحقيقية تعكس جودة البيانات وليس خطأ في التمثيل.

Before storing real data in data structures, test the project's structure on small synthetic data. Create a dictionary containing 3-4 mock samples, and verify that it works as expected. Then try adding new data, deleting data, and accessing data via identifiers. If these tests succeed, the workflow is confident that the project's structure is correct, and any problem with real data reflects data quality, not a representation error.

Chapter 8 — Functions, Modules, and Error Handling

الفصل 8 — الدوال والوحدات وإدارة الأخطاء

Deeper Conceptual Explanation

الفكرة المركزية: functions، modules، exceptions، validation. الفشل المبكر.

شرح أعمق للمفهوم

A good function defines a clear contract: what enters it, what it returns, and which states it rejects. Error handling is not about hiding problems; it turns ambiguous failure into a state that can be logged, tested, and reviewed. In scientific software, a clear failure message can be more valuable than a silent result that merely looks successful.

الدالة الجيدة تحدد عقداً واضحاً: ما الذي يدخل إليها، وما الذي تخرجه، وما الحالات التي ترفضها. إدارة الأخطاء ليست إخفاء المشكلة؛ بل تحويل الفشل الغامض إلى حالة يمكن تسجيلها واختبارها ومراجعتها. في مشروع علمي، الرسالة الواضحة عند الفشل قد تكون أهم من نتيجة صامتة تبدو ناجحة.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. الدوال (Functions)

1. Functions

A function is a block of code that performs a specific task. the workflow can call it multiple times from different places in the project's program, reducing repetition and making code more readable and testable. In bioinformatics, functions are how you transform analysis steps into independent units: a function to read a file, a function to validate a sequence, a function to calculate GC content, and so on.

الدالة هي كتلة من الكود تؤدي مهمة محددة. يمكن استدعاؤها عدة مرات من أماكن مختلفة في برنامجك، مما يقلل التكرار ويجعل الكود أكثر قابلية للقراءة والاختبار. في المعلوماتية الحيوية، الدوال هي الطريقة التي تحوّل بها خطوات التحليل إلى وحدات مستقلة: دالة لقراءة الملف، دالة للتحقق من صحة التسلسل، دالة لحساب نسبة GC، وهكذا.

A good function does not merely perform a task; it clearly defines its expected inputs, outputs, and what constitutes an invalid state. This clarity is what makes code testable. If the project's function accepts anything and produces anything, you do not know when or why it fails.

الدالة الجيدة لا تنفذ مهمة فقط؛ بل تحدد بوضوح مدخلاتها المتوقعة، ومخرجاتها، وما يشكل حالة غير صالحة. هذا الوضوح هو ما يجعل الكود قابلاً للاختبار. إذا كانت دالتك تقبل أي شيء وتنتج أي شيء، فأنت لا تعرف متى تفشل أو لماذا.

```
# دالة بسيطة للتحقق من صحة تسلسل DNA
def validate_dna(sequence):
    """تتحقق من أن التسلسل يحتوي على الحروف A، T، C، G فقط."""
    valid_bases = set("ATCG")
    if not set(sequence.upper()).issubset(valid_bases):
        raise ValueError(f"التسلسل يحتوي على أحرف غير صالحة: {sequence}")
    return sequence.upper()
# استدعاء الدالة
validate_dna("ATGC")  # يعمل
validate_dna("ATNX")  # يرفع خطأ ValueError
```

الدوال المعرفة هنا (validate_dna) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها.

Notice that the function clearly defines what is valid input and what is not, and raises a meaningful error when it encounters invalid input. This is the essence of "fail early": detect the error at the earliest possible point, rather than letting it spread to other parts of the program.

لاحظ أن الدالة تحدد بوضوح ما هو مدخل صالح وما هو غير صالح، وترفع خطأً معنّىً عندما تواجه مدخلاً غير صالح. هذا هو جوهر "الفشل المبكر" (fail early): اكتشف الخطأ في أقرب نقطة ممكنة، بدلاً من تركه ينتشر إلى أجزاء أخرى من البرنامج.

2. الوحدات (Modules)

2. Modules

A module is a file containing related functions and variables. Instead of writing everything in one large file, the workflow can organize the project's functions into modules by topic: a module for sequence handling, a module for file handling, a module for statistical analysis. This organization makes code easier to understand, test, and reuse.

In bioinformatics, using modules reflects the way you organize scientific experiments: each step has its place, and each tool has its role. When you return to a project months later, or when a colleague reads the project's code, organized modules make it easy to follow the logic.

الوحدة هي ملف يحتوي على دوال ومتغيرات مرتبطة ببعضها. بدلاً من كتابة كل شيء في ملف واحد كبير، يمكن تنظيم دوالك في وحدات حسب الموضوع: وحدة للتعامل مع التسلسلات، وحدة للتعامل مع الملفات، وحدة للتحليل الإحصائي. هذا التنظيم يجعل الكود أسهل في الفهم والاختبار وإعادة الاستخدام.

في المعلوماتية الحيوية، استخدام الوحدات يعكس الطريقة التي تنظّم بها التجارب العلمية: كل خطوة لها مكانها، وكل أداة لها دورها. عندما تعود إلى مشروع بعد أشهر، أو عندما يقرأ زميل كودك، فإن الوحدات المنظمة تجعل من السهل تتبع المنطق.

```
def gc_content(seq):
    # حساب نسبة GC
    seq = seq.upper()
    gc_count = seq.count('G') + seq.count('C')
    return gc_count / len(seq) * 100
#في ملف sequence_utils.py
def validate_dna(seq):
    # ... الكود السابق ...
    pass
```

```
#في ملف آخر، يمكنك استيراد الدوال
from sequence_utils import validate_dna, gc_content
```

3. Error Handling (Exceptions)

In biological analysis, inputs are rarely perfect. A file may be missing, a sequence may contain unexpected characters, or values may be absent. Instead of the program crashing suddenly (which happens if you do not handle errors), the workflow can use try and except to handle errors gracefully.

3. إدارة الأخطاء (Exceptions)

From a scientific perspective, failures are not just technical glitches; they are signals that an assumption in the analysis may not be met. A missing file means the path is wrong or the file was never created. A mismatched length means the data does not match expectations. An invalid character in a sequence means the data may be corrupted or from a different format. Every error is evidence of what needs to be checked or fixed.

في التحليل البيولوجي، المدخلات نادراً ما تكون مثالية. قد يكون الملف ناقصاً، أو التسلسل يحتوي على أحرف غير متوقعة، أو القيم مفقودة. بدلاً من أن ينهار البرنامج فجأة (وهو ما يحدث إذا لم تتعامل مع الأخطاء)، يمكن استخدام try و except للتعامل مع الأخطاء بشكل منظم.

من منظور علمي، حالات الفشل ليست مجرد أعطال تقنية؛ هي إشارات إلى أن افتراضاً من افتراضات التحليل قد لا يكون متحققاً. ملف غير موجود يعني أن المسار خاطئ أو أن الملف لم يُنشأ. طول غير متوافق يعني أن البيانات لا تتطابق مع التوقعات. حرف غير صالح في التسلسل يعني أن البيانات قد تكون تالفة أو من صيغة مختلفة. كل خطأ هو دليل على ما يجب التحقق منه أو إصلاحه.

Purpose: read or write a file with an explicit input or output path as part of reproducibility.

الغرض: قراءة أو كتابة ملف مع تحديد مسار الإدخال أو الإخراج بوضوح، وهو جزء من قابلية إعادة الإنتاج.

```
try:
    with open("data.fasta", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("الملف غير موجود. تحقق من المسار.")
except PermissionError:
    print("لا تملك صلاحية قراءة هذا الملف.")
except Exception as e:
    print(f"حدث خطأ غير متوقع: {e}")
```

This code handles specific failure cases separately and gives clear messages for each case. This is the distinction between a technical error (such as a syntax error) and an input error (such as a missing file). Each type requires different handling.

4. Defensive Programming as Quality Control

هذا الكود يتعامل مع حالات فشل محددة بشكل منفصل، ويعطي رسائل واضحة لكل حالة. هذا هو التمييز بين الخطأ التقني (مثل خطأ في بناء الجملة) والخطأ في المدخلات (مثل ملف ناقص). كل نوع يحتاج إلى معالجة مختلفة.

Defensive programming is the practice of writing code that assumes everything can fail, and validates everything before using it. In bioinformatics, this is like quality control in the laboratory: before you use a sample, you check its identity, date, and condition. Similarly, before you use a function, it is appropriate to validate its inputs. This transformation of defensive programming into quality control is what makes scientific code reliable.

4. البرمجة الدفاعية كمراقبة جودة

البرمجة الدفاعية هي ممارسة كتابة كود يفترض أن كل شيء قد يفشل، ويتحقق من صحة كل شيء قبل استخدامه. في المعلوماتية الحيوية، هذا يشبه مراقبة الجودة في المختبر: قبل أن تستخدم عينة، تتحقق من هويتها، وتاريخها، وحالتها. وبالمثل، قبل أن تستخدم دالة، يجب أن تتحقق من مدخلاتها. هذا التحويل للبرمجة الدفاعية إلى مراقبة جودة هو ما يجعل الكود العلمي موثوقاً.

```
gc = seq.count('G') + seq.count('C')
return gc / len(seq) * 100
def safe_gc_content(sequence):
"""مع تحقق مسبق من صحة التسلسل GC تحسب نسبة."""
try:
seq = validate_dna(sequence)
except ValueError as e:
print(f"خطأ في المدخلات: {e}")
return None  # أو إعادة قيمة افتراضية معقولة
```

الدوال المعرفة هنا (safe_gc_content) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

5. Practical Application

5. التطبيق العملي

Let us build an integrated example: you want to analyze several FASTA files, calculate GC content for each, and store the results in a dictionary. Using functions, modules, and error handling.

دعنا نبني مثالاً متكاملاً: أردت تحليل عدة ملفات FASTA، وحساب نسبة GC لكل منها، وتخزين النتائج في قاموس.

باستخدام الدوال والوحدات وإدارة الأخطاء:

Purpose: parse a sequence file into processable objects using the specified file format.

الغرض: تحويل ملف تسلسل إلى كائنات قابلة للمعالجة باستخدام الصيغة المحددة في القراءة.

```
import os
from Bio import SeqIO
from sequence_utils import validate_dna, gc_content
def analyze_fasta_files(directory):
  """تحليل جميع ملفات FASTA في مجلد معين."""
  results = {}
for filename in os.listdir(directory):
    if not filename.endswith(".fasta"):
```

continue

```
filepath = os.path.join(directory, filename)
    try:
      for record in SeqIO.parse(filepath, "fasta"):
        seq_str = str(record.seq)
        try:
          validated_seq = validate_dna(seq_str)
          gc = gc_content(validated_seq)
```

results[record.id] = {"gc": gc, "length": len(validated_seq)}

```
        except ValueError as e:
          print(f"خطأ في تسلسل {record.id} في ملف {filename}: {e}")
    except FileNotFoundError:
      print(f"الملف غير موجود: {filepath}")
    except Exception as e:
      print(f"خطأ غير متوقع في ملف {filename}: {e}")
return results
```

وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

This code brings together everything we have learned: specific functions, organized modules, and comprehensive error handling. Each error is identified and handled separately, and the final result remains usable even if some files fail.

هذا الكود يجمع كل ما تعلمناه: دوال محددة، وحدات منظمة، وإدارة أخطاء شاملة. كل خطأ يتم التعرف عليه ومعالجته بشكل منفصل، والنتيجة النهائية لا تزال قابلة للاستخدام حتى لو فشل بعض الملفات.

Chapter Summary

خلاصة الفصل

Functions, modules, and error handling are the tools that transform code from a sequence of commands into an organized scientific system. Functions make operations testable, modules make code organizable, and error handling makes it capable of dealing with the reality of imperfect data. In bioinformatics, these tools are not luxuries; they are necessities. The next chapter will build on this foundation to introduce you to Object-Oriented Programming (OOP) as a way to represent biological entities more richly and integratively.

الدوال والوحدات وإدارة الأخطاء هي الأدوات التي تحوّل الكود من مجموعة أوامر متسلسلة إلى نظام علمي منظم. الدوال تجعل العمليات قابلة للاختبار، والوحدات تجعل الكود قابلاً للتنظيم، وإدارة الأخطاء تجعله قادراً على التعامل مع واقع البيانات غير المثالية. في المعلوماتية الحيوية، هذه الأدوات ليست رفاهية؛ إنها ضرورة. الفصل القادم سيبني على هذا الأساس ليقدم لك البرمجة الكائنية (OOP) كطريقة لتمثيل الكيانات البيولوجية بشكل أكثر غنى وتكاملاً.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Functions, Modules, and Error Handling" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "الدوال والوحدات وإدارة الأخطاء" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "using functions guarantees the correctness of biological results." This is incorrect; functions organize code and make it testable, but they do not verify the validity of biological hypotheses. The correct statement is: "functions, modules, and error handling make code organized and testable, and allow early detection of errors, but the correctness of biological results depends on data quality and sound analysis." A simple test: if the project’s function raises an error, does it tell you about a specific problem in the input, or just a generic messag?e.

مثال على التجاوز: القول إن "استخدام الدوال يضمن صحة النتائج البيولوجية". هذا غير صحيح؛ الدوال تنظم الكود وتجعله قابلاً للاختبار، لكنها لا تتحقق من صحة الفرضيات البيولوجية. الصحيح هو: "الدوال والوحدات وإدارة الأخطاء تجعل الكود منظمًا وقابلاً للاختبار، وتسمح باكتشاف الأخطاء مبكراً، لكن صحة النتائج البيولوجية تعتمد على جودة البيانات وسلامة التحليل." اختبار بسيط: إذا كانت دالتك ترفع خطأً، فهل يخبرك هذا الخطأ بمشكلة محددة في المدخلات أم مجرد رسالة عامة؟

Example: A function that validates a sequence and raises an error if it contains invalid characters. This is a computational representation. Claiming that "this sequence is biologically invalid" is a biological claim that goes beyond what the function tests. The function only tests that the sequence contains no unexpected characters; it does not test whether the sequence represents a real gene or functional region. Practical test: before you infer something biological from a function's result, ensure there is context and additional evidence supporting the project’s conclusion.

مثال: دالة تتحقق من صحة التسلسل وترفع خطأً إذا احتوى على أحرف غير صالحة. هذا هو تمثيل حسابي. الادعاء بأن "هذا التسلسل غير صالح بيولوجياً" هو ادعاء بيولوجي يتجاوز ما تختبره الدالة. الدالة تختبر فقط أن التسلسل لا يحتوي على أحرف غير متوقعة؛ لا تختبر ما إذا كان التسلسل يمثل جيناً حقيقياً أو منطقة وظيفية. اختبار عملي: قبل أن تستنتج شيئاً بيولوجياً من نتيجة دالة، تأكد من وجود سياق وأدلة إضافية تدعم استنتاجك.

مثال تطبيقي على دالة تقرأ ملف FASTA وتحسب نسبة GC:

· المدخلات: مسار ملف FASTA.

· الافتراضات: أن الملف موجود وقابل للقراءة، وأنه يحتوي على تسلسلات بصيغة FASTA صحيحة، وأن التسلسلات لا تحتوي على أحرف غير صالحة.

· المخرجات: قاموس يربط معرف كل تسلسل بنسبة GC الخاصة به.

· القيود: الدالة لا تتحقق من جودة التسلسل، ولا تتعامل مع الملفات المضغوطة، ولا تحلل البيانات الوصفية.

Applied example of a function that reads a FASTA file and calculates GC content.

اختبار: قبل تشغيل الدالة على مئات الملفات، اختبرها على ملف واحد صغير معروف المحتوى، وتأكد من أنها تعطي النتيجة المتوقعة.

· Inputs: path to a FASTA file

· Assumptions: that the file exists and is readable, that it contains sequences in correct FASTA format, and that the sequences contain no invalid characters.

· Outputs: a dictionary linking each sequence identifier to its GC content.

· Limitations: the function does not check sequence quality, does not handle compressed files, and does not parse metadata.

Test: before running the function on hundreds of files, test it on one small file with known content, and verify that it gives the expected result.

عند كتابة دوال للتحليل، يجب تسجيل:

· المصدر: أي بيانات استخدمتها لتطوير الدالة (مثل ملفات اختبار).

· الإصدار: إصدار Python، وأي مكتبات استخدمتها (مثل Biopython).

· المعلمات: أي اختيارات قمت بها في تصميم الدالة (مثل العتبات، أو الطرق الافتراضية).

· التحويلات: أي خطوات قمت بها لتنظيف البيانات أو تحويلها قبل استخدامها في الدالة.

When writing analysis functions, record.

اختبار: فحص تطبيقي: "لو أعطيت هذه الدالة لزميل بعد عام، هل يستطيع فهم ما تفعله ولماذا، ويعيد استخدامها في مشروعه؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: any data used to develop the function (e.g., test files).

· Version: Python version, and any libraries used (e.g., Biopython).

· Parameters: any choices made in function design (e.g., thresholds, default methods).

· Transformations: any steps taken to clean or transform data before using it in the function.

Test: Practical check: "If I gave this function to a colleague a year later, could they understand what it does and why, and reuse it in their project?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على دوالك في تحليل بيانات حقيقية، اختبر كل دالة على بيانات اصطناعية صغيرة. اكتب اختبارات تتحقق من:

· أن الدالة تعمل مع مدخلات صحيحة.

· أن الدالة ترفع الخطأ المناسب مع مدخلات غير صحيحة.

· أن الدالة تتعامل مع الحالات الحدية (مثل تسلسل فارغ، أو ملف فارغ، أو مسار غير موجود).

Before relying on the project’s functions to analyze real data, test each function on small synthetic data. Write tests that check.

إذا نجحت هذه الاختبارات، فأنت واثق من أن دوالك تعمل كما هو متوقع، وأي مشكلة في البيانات الحقيقية تعكس جودة البيانات وليس خطأ في الكود.

· That the function works with correct inputs.

· That the function raises the appropriate error with incorrect inputs.

· That the function handles edge cases (e.g., empty sequence, empty file, or non-existent path).

If these tests succeed, the workflow is confident that the project's functions work as expected, and any problem with real data reflects data quality, not a code error.

Chapter 9 — Object-Oriented Programming

الفصل 9 — البرمجة الكائنية

الفكرة المركزية: objects، encapsulation، composition، inheritance، قابلية الاختبار.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

البرمجة الكائنية تصبح مفيدة هنا عندما ترتبط ببنية البيانات البيولوجية نفسها. Seq يمثل التسلسل، وSeqRecord يضيف الهوية والوصف، وSeqFeature يضيف موقعاً وميزة. فصل هذه المستويات يجعل البرنامج أقرب إلى بنية السجل العلمي، ويقلل الحاجة إلى متغيرات منفصلة قد تفقد العلاقة بينها.

Object-oriented programming becomes useful here because it mirrors the structure of biological data. Seq represents the sequence, SeqRecord adds identity and description, and SeqFeature adds a location and feature. Keeping these levels explicit makes the program closer to a scientific record and reduces reliance on disconnected variables whose relationships can be lost.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is Object-Oriented Programming?

1. ما هي البرمجة الكائنية؟

Object-Oriented Programming is a programming paradigm that organizes code around "objects" rather than separate functions and data. An object is a unit that contains data (called Attributes or Properties) and behavior (called Methods). Imagine an object as a small factory: it has inputs (data), internal processes (methods), and outputs (results). Instead of passing data to an external function, you ask the object itself to perform the operation. This model makes you think about "who does what" rather than "what happens to the dat"a.

البرمجة الكائنية هي نموذج برمجي ينظم الكود حول "الكائنات" (Objects) بدلاً من الدوال والبيانات المنفصلة. الكائن هو وحدة تحتوي على بيانات (تُسمى Attributes أو Properties) وسلوك (تُسمى Methods). تخيل أن الكائن هو مصنع صغير: له مدخلات (البيانات)، وعمليات داخلية (الطرق)، ومخرجات (النتائج). بدلاً من أن تمرر

البيانات إلى دالة خارجية، تطلب من الكائن نفسه أن يقوم بالعملية. هذا النموذج يجعلك تفكر في "من يفعل ماذا" بدلاً من "ماذا يحدث للبيانات"ـ

In bioinformatics, this model is very natural. A DNA sequence is not just text; it has properties (length, orientation, GC content) and behaviors (complement, translation, alignment). The Seq object in Biopython is an embodiment of this idea: you give it a sequence, and the object knows how to handle itself.

في المعلوماتية الحيوية، هذا النموذج طبيعي جداً. فالتسلسل DNA ليس مجرد نص؛ له خصائص (الطول، الاتجاه، المحتوى GC) وسلوكيات (التكامل، الترجمة، المحاذاة). الكائن Seq في Biopython هو تجسيد لهذه الفكرة: تأخذ تسلسلاً، والكائن يعرف كيف يتعامل مع نفسهـ

2. Fundamental Concepts

2. المفاهيم الأساسية

Encapsulation: is the bundling of data and the operations that process it within a single object, hiding internal details from the outside world. The object presents an interface (methods) to interact with it, but hides how it works internally. In a biological context, encapsulation means that the Seq object knows how to compute the reverse complement, but you do not care how it does it; what matters is that you call reverse_complement() and get the correct result.

التغليف (Encapsulation) : هو جمع البيانات والعمليات التي تعالجها داخل كائن واحد، وإخفاء التفاصيل الداخلية عن العالم الخارجي. الكائن يقدم واجهة (methods) للتفاعل معه، لكنه يخفي كيف يعمل داخلياً. في السياق البيولوجي، التغليف يعني أن الكائن Seq يعرف كيف يحسب المتمم العكسي، لكن لا يهمك كيف يفعل ذلك؛ المهم أن تستدعي reverse_complement() وتحصل على النتيجة الصحيحةـ

Composition: is building a complex object from simpler objects. In Biopython, a SeqRecord object contains a Seq object (the sequence) and SeqFeature objects. This composition reflects the natural structure of biological data: a genetic record contains a sequence and additional information. Composition is the preferred approach in Biopython over inheritance, because it is more flexible and easier to understand.

التكوين (Composition) : هو بناء كائن معقد من كائنات أبسط. في Biopython، كائن SeqRecord يحتوي على كائن Seq (التسلسل) وميزات SeqFeature. هذا التكوين يعكس البنية الطبيعية للبيانات البيولوجية: سجل جيني يحتوي على تسلسل ومعلومات إضافية. التكوين هو الأسلوب المفضل في Biopython عن الوراثة، لأنه أكثر مرونة وأسهل في الفهمـ

Inheritance: is a mechanism that allows an object (the derived class) to inherit properties and behaviors from another object (the base class). Inheritance is useful when the workflow has several types of objects that share basic behavior, but each has differences. In Biopython, inheritance is not as common as composition. Most objects are built through composition rather than inheritance. This is an important lesson: inheritance is not always the best solution.

الوراثة (Inheritance) : هي آلية تسمح لكائن (الفئة المشتقة) أن يرث خصائص وسلوكيات كائن آخر (الفئة الأساسية). الوراثة مفيدة عندما يكون يوجد في المثال عدة أنواع من كائنات تشترك في سلوك أساسي، لكن لكل منها اختلافات. في Biopython، الوراثة ليست شائعة مثل التكوين. معظم الكائنات تُبنى بالتكوين بدلاً من الوراثة. هذا درس مهم: الوراثة ليست دائماً الحل الأفضل.

Polymorphism: is the ability of different objects to respond to the same message (the same function name) in their own way. In Biopython, SeqIO is an example of polymorphism: you use the same functions (read(), parse()) to read multiple file formats (FASTA, GenBank, and others). The code does not care about the file type; the objects know how to handle themselves.

تعدد الأشكال (Polymorphism) : هو قدرة كائنات مختلفة على الاستجابة لنفس الرسالة (نفس اسم الدالة) بطريقتها الخاصة. في Biopython، SeqIO هو مثال على تعدد الأشكال: تستخدم نفس الدوال (read()، parse()) لقراءة صيغ متعددة من الملفات (FASTA، GenBank، وغيرهما). الكود لا يهتم بنوع الملف؛ الكائنات تعرف كيف تتعامل مع نفسها.

3. Why Is Composition Better Than Inheritance in Bioinformatics?

3. لماذا التكوين أفضل من الوراثة في المعلوماتية الحيوية؟

In bioinformatics, biological entities tend to be composed of parts, rather than being specializations of a single type. A gene contains a sequence, features, and metadata. A cell contains components. Composition reflects this naturally: an object has other objects, rather than being a type of another object. Inheritance, on the other hand, assumes an "is-a" relationship: a cat is a type of animal. But a sequence is not a type of record; a record contains a sequence. So composition (has-a) is more appropriate. In Biopython, you see this clearly: SeqRecord contains a Seq, rather than inheriting from it.

في المعلوماتية الحيوية، تميل الكيانات البيولوجية إلى أن تكون مركبة من أجزاء، وليس تخصصات من نوع واحد. الجين يحتوي على تسلسل، وميزات، وبيانات وصفية. الخلية تحتوي على مكونات. التكوين يعكس هذا بشكل طبيعي: الكائن يمتلك كائنات أخرى، وليس هو نوعاً من نوع آخر. الوراثة، من ناحية أخرى، تفترض علاقة "هو" (is-a): القطة

هي نوع من الحيوان. لكن التسلسل ليس نوعاً من السجل؛ السجل يحتوي على تسلسل. لذا التكوين (has-a) هو الأنسب. في Biopython، ترى هذا بوضوح: SeqRecord يحتوي على Seq، وليس يرث منه.

```
# التكوين: SeqRecord يحتوي على Seq
class SeqRecord:
    def __init__(self, seq, id, description):
        self.seq = seq          # seq هو كائن Seq
        self.id = id
        self.description = description
        self.features = []      # قائمة من SeqFeature
```

الدوال المعرفة هنا (__init__) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (SeqRecord) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

Read this example in three layers: inputs, the transformation performed by functions or objects, and the resulting output or checks. The class-based structure keeps data and behavior together so the same tested logic can be reused without duplicating the analysis.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة SeqRecord يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

4. Practical Example: Building a Simple Biological Object

Let us build a simple Gene object that brings together what we have learned.

4. مثال عملي: بناء كائن بيولوجي بسيط

دعنا نبني كائن Gene بسيطاً يجمع ما تعلمناه:

Purpose: apply base complementarity and then reverse orientation to obtain the reverse complement.

الغرض: تطبيق قاعدة التكامل القاعدي ثم عكس الاتجاه للحصول على المتمم العكسي.

```
class Gene:
    """يمثل جيناً بسيطاً مع تسلسل وهوية وبيانات وصفية."""
    def __init__(self, gene_id, sequence, organism="unknown"):
        self.gene_id = gene_id
        self.organism = organism
        self.sequence = sequence.upper()
        self._validate_sequence()
    def gc_content(self):
        """حساب نسبة GC."""
        gc_count = self.sequence.count('G') + self.sequence.count('C')
        return gc_count / len(self.sequence) * 100
    def _validate_sequence(self):
        """التحقق من صحة التسلسل (طريقة داخلية)."""
        valid_bases = set("ATCG")
        if not set(self.sequence).issubset(valid_bases):
            raise ValueError(f"تسلسل غير صالح للجين {self.gene_id}")
    def reverse_complement(self):
        """حساب المتمم العكسي (محاكاة لـ Biopython)."""
        complement = {'A': 'T', 'T': 'A', 'C': 'G', 'G': 'C'}
        return ''.join(complement[base] for base in reversed(self.sequence))
    def __repr__(self):
        return f"Gene({self.gene_id}, {self.organism}, length={len(self.sequence)})"
# استخدام الكائن
gene = Gene("BRCA1", "ATGCGTACGTAGCTAGCTAGCTAGCTAGC", "human")
print(gene.gc_content())        # نسبة GC
print(gene.reverse_complement()) # المتمم العكسي
print(gene)                      # تمثيل الكائن
```

الدوال المعرفة هنا (__init__, _validate_sequence, gc_content, reverse_complement, __repr__) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (Gene) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها.

في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة Gene يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

لاحظ أن:

· التغليف: البيانات (gene_id، sequence) والسلوك (gc_content، reverse_complement) مجمعة في كائن واحد.

Notice that

· التحقق من الصحة: دالة _validate_sequence تتحقق من المدخلات عند إنشاء الكائن (فشل مبكر).

· Encapsulation: data (gene_id, sequence) and behavior (gc_content, reverse_complement) are bundled in one object.

· التكوين: هذا الكائن يمكن أن يحتوي على كائنات أخرى (مثل قائمة من الميزات).

· Validation: the _validate_sequence method checks inputs when the object is created (fail early).

· Composition: this object can contain other objects (e.g., a list of features).

5. The Relationship with Biopython

5. العلاقة مع Biopython

What we built here is a simple simulation of what Biopython does. In Biopython, the Seq object provides basic operations (such as complement(), reverse_complement(), and translate()). The SeqRecord object combines Seq with identity and metadata. The SeqFeature object represents localized features. Understanding OOP helps you understand how Biopython works internally, why it is designed this way, and how the workflow can extend it for the project's own projects.

ما بنيناه هنا هو محاكاة بسيطة لما يفعله Biopython. في Biopython، كائن Seq يقدم العمليات الأساسية (مثل complement() و reverse_complement() و translate()). كائن SeqRecord يجمع Seq مع

الهوية والبيانات الوصفية. كائن SeqFeature يمثل الميزات الموضعية. فهم البرمجة الكائنية يساعدك على فهم كيف يعمل Biopython من الداخل، ولماذا صُمم بهذه الطريقة، وكيف يمكن تمديده لمشاريعك الخاصة.

6. Practical Application: Testing the Object

Before using this object in a real project, test it on synthetic data.

6. التطبيق العملي: اختبار الكائن

قبل أن تستخدم هذا الكائن في مشروع حقيقي، اختبره على بيانات اصطناعية:

```
# اختبارات بسيطة
def test_gene():
    # إنشاء جين صحيح
    gene = Gene("TEST1", "ATGC", "human")
    assert gene.gc_content() == 50.0
    assert gene.reverse_complement() == "GCAT"
# اختبار المدخلات غير الصالحة
    try:
        bad_gene = Gene("TEST2", "ATNX")
    except ValueError as e:
        print(f"تم اكتشاف خطأ كما هو متوقع: {e}")
test_gene()
```

الدوال المعرفة هنا (test_gene) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

Chapter Summary

خلاصة الفصل

Object-Oriented Programming is a way to organize code around objects that combine data and behavior together. The fundamental concepts — encapsulation, composition, inheritance, and polymorphism — give you powerful tools to build software systems that reflect the natural structure of biological data. In Biopython, composition is the dominant approach, where SeqRecord contains Seq and features, rather than inheriting from them. Understanding these concepts enables you to read Biopython code with deeper insight, and to write the project's own code in an organized and testable way. The next chapter will take you into the heart of Biopython with the Seq object.

البرمجة الكائنية هي طريقة لتنظيم الكود حول الكائنات التي تجمع البيانات والسلوك معاً. المفاهيم الأساسية — التغليف، التكوين، الوراثة، وتعدد الأشكال — تمنحك أدوات قوية لبناء أنظمة برمجية تعكس البنية الطبيعية للبيانات البيولوجية. في Biopython، التكوين هو الأسلوب السائد، حيث يحتوي SeqRecord على Seq وميزات، بدلاً من أن يرث منها. فهم هذه المفاهيم يجعلك قادراً على قراءة كود Biopython بفهم أعمق، وكتابة كودك الخاص بطريقة منظمة وقابلة للاختبار. الفصل القادم سيدخلك إلى قلب Biopython مع كائن Seq.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Object-Oriented Programming" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "البرمجة الكائنية" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "using OOP guarantees the correctness of biological models." This is incorrect; OOP organizes code, but it does not prove the correctness of biological models. The correct statement is: "OOP makes code more organized, understandable, and testable, which helps in building computational models that reflect biological structure, but the correctness of the biological model depends on the data and hypotheses." A simple test: when you design a biological object, Practical check: "Does this object reflect the structure of the data I have, or does it impose an unnatural structure on i?"t.

مثال على التجاوز: القول إن "استخدام البرمجة الكائنية يضمن صحة النماذج البيولوجية". هذا غير صحيح؛ البرمجة الكائنية تنظم الكود، لكنها لا تثبت صحة النماذج البيولوجية. الصحيح هو: "البرمجة الكائنية تجعل الكود أكثر تنظيماً وقابلية للفهم والاختبار، مما يساعد في بناء نماذج حاسوبية تعكس البنية البيولوجية، لكن صحة النموذج البيولوجي تعتمد على البيانات والفرضيات." اختبار بسيط: عندما تصمم كائناً بيولوجياً، فحص تطبيقي: "هل يعكس هذا الكائن بنية البيانات التي أملكها، أم أنه يفرض هيكلاً غير طبيعي عليها؟".

Example: Designing a Gene object with methods like gc_content() and reverse_complement() is a computational representation. Claiming that "this object represents a real gene in the cell" is a biological claim not proven by the object. The object knows how to calculate sequence properties, but it does not know whether this sequence expresses a real gene, or whether it is expressed in a specific tissue. Practical test: before you consider that an object represents a biological entity, ensure that the project’s data and context support this representation.

مثال: تصميم كائن Gene بطرق مثل gc_content() و reverse_complement() هو تمثيل حسابي. الادعاء بأن "هذا الكائن يمثل جيناً حقيقياً في الخلية" هو ادعاء بيولوجي لا يثبته الكائن. الكائن يعرف كيف يحسب خصائص التسلسل، لكنه لا يعرف إذا كان هذا التسلسل يعبر عن جين حقيقي، أو إذا كان معبراً عنه في نسيج معين. اختبار عملي: قبل أن تعتبر أن كائناً يمثل كياناً بيولوجياً، تأكد من أن بياناتك وسياقك يدعمان هذا التمثيل.

مثال تطبيقي على كائن Gene:

· المدخلات: معرف الجين، التسلسل (نصي)، الكائن الحي (اختياري).

· الافتراضات: أن التسلسل يحتوي فقط على الحروف A، T، C، G (أو صيغة مماثلة)، وأن المعرف فريد.

· المخرجات: كائن Gene يمكنه حساب نسبة GC، المتمم العكسي، وغيرها.

· القيود: الكائن لا يتحقق من صحة المعرف، ولا يتعامل مع تسلسلات بروتين، ولا يحلل الميزات الجينية مثل الإكسونات والإنترونات.

Applied example of a Gene object.

اختبار: قبل تصميم كائن، حدد بوضوح ما يمثله، وما لا يمثله. هذا يمنع استخدامه في سياقات غير مناسبة.

· Inputs: gene identifier, sequence (text), organism (optional).

· Assumptions: that the sequence contains only A, T, C, G (or equivalent), and that the identifier is unique.

· Outputs: a Gene object that can calculate GC content, reverse complement, and others.

· Limitations: the object does not validate the identifier, does not handle protein sequences, and does not analyze genetic features like exons and introns.

Test: before designing an object, clearly define what it represents and what it does not. This prevents using it in inappropriate contexts.

عند تصميم كائنات للتحليل البيولوجي، يجب تسجيل:

· المصدر: البيانات التي استخدمتها لتصميم الكائن.

· الإصدار: إصدار Python وأي مكتبات استخدمتها.

· المعلمات: أي اختيارات في تصميم الكائن (مثل كيفية التعامل مع الحروف الصغيرة والكبيرة).

· التحويلات: أي خطوات قمت بها لتحويل البيانات الخام إلى شكل الكائن.

When designing objects for biological analysis, record.

اختبار: فحص تطبيقي: "لو أعطيت هذا الكائن لزميل بعد عام، هل يستطيع فهم ما يمثله وكيف يستخدمه، ويعيد استخدامه في مشروعه دون تعديلات جوهرية؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the data used to design the object.

· Version: Python version and any libraries used.

· Parameters: any choices in object design (e.g., how to handle uppercase/lowercase).

· Transformations: any steps taken to transform raw data into the object's form.

Test: Practical check: "If I gave this object to a colleague a year later, could they understand what it represents and how to use it, and reuse it in their project without major modifications?" If the answer is no, the documentation is insufficient.

قبل أن تستخدم الكائنات التي تصممها على بيانات حقيقية، اختبرها على بيانات اصطناعية. اكتب اختبارات تتحقق من:

· أن الكائن يُنشأ بشكل صحيح مع مدخلات صحيحة.

· أن الكائن يرفع الأخطاء المناسبة مع مدخلات غير صحيحة.

· أن جميع الطرق (methods) تعطي النتائج المتوقعة على بيانات معروفة.

Before using the objects you design on real data, test them on synthetic data. Write tests that check.

إذا نجحت هذه الاختبارات، فأنت واثق من أن الكائن يعمل كما هو متوقع.

· That the object is created correctly with valid inputs.

· That the object raises appropriate errors with invalid inputs.

· That all methods give the expected results on known data.

If these tests succeed, the workflow is confident that the object works as expected.

A realistic OOP application: objects for reads and sample QC

تطبيق OOP واقعي: كائنات تمثل القراءة وجودة العينة

OOP becomes genuinely useful when a biological entity has state, behavior, and validation rules. The following example goes beyond a symbolic Gene class: FastqRecord represents one read, FastqQC represents accumulated quality information, and iter_fastq handles compressed or uncompressed input. It remains educational, but its architecture resembles a real utility: validate inputs, separate responsibilities, and emit a testable report.

البرمجة الكائنية تصبح ذات قيمة حقيقية عندما يكون لدينا كيان بيولوجي له حالة وسلوك وقواعد تحقق. المثال التالي لا يكتفي بإنشاء فئة رمزية باسم Gene؛ بل يبني طبقة صغيرة لمعالجة FASTQ: تمثل FastqRecord قراءة واحدة، وتمثل FastqQC حالة تقرير الجودة، وتنفذ iter_fastq عملية القراءة من ملف مضغوط أو غير مضغوط. الهدف تعليمي، لكنه قريب من البنية التي يحتاجها برنامج حقيقي: التحقق من المدخل، فصل المسؤوليات، وإخراج تقرير يمكن اختباره.

```
from dataclasses import dataclass, field
from pathlib import Path
from typing import Iterable, Iterator
import gzip
VALID_BASES = set("ACGTN")
@dataclass(frozen=True)
class FastqRecord:
    identifier: str
    sequence: str
    quality: str
def validate(self) -> None:
        if not self.identifier:
            raise ValueError("FASTQ identifier is empty")
        if len(self.sequence) != len(self.quality):
            raise ValueError("Sequence and quality lengths differ")
        invalid = set(self.sequence.upper()) - VALID_BASES
        if invalid:
            raise ValueError(f"Unexpected bases: {sorted(invalid)}")
@property
    def length(self) -> int:
        return len(self.sequence)
@property
    def gc_fraction(self) -> float:
        seq = self.sequence.upper()
        if not seq:
            return 0.0
        return (seq.count("G") + seq.count("C")) / len(seq)
def phred_scores(self) -> list[int]:
        return [ord(char) - 33 for char in self.quality]
def mean_quality(self) -> float:
        scores = self.phred_scores()
        return sum(scores) / len(scores) if scores else 0.0
@dataclass
class FastqQC:
    total_records: int = 0
    invalid_records: int = 0
    total_bases: int = 0
    gc_sum: float = 0.0
    duplicate_ids: set[str] = field(default_factory=set)
def add(self, record: FastqRecord, seen_ids: set[str]) -> None:
        self.total_records += 1
        self.total_bases += record.length
        self.gc_sum += record.gc_fraction
if record.identifier in seen_ids:
            self.duplicate_ids.add(record.identifier)
        seen_ids.add(record.identifier)
```

```
def summary(self) -> dict:
        mean_gc = self.gc_sum / self.total_records if self.total_records else 0.0
        return {
            "records": self.total_records,
            "bases": self.total_bases,
            "mean_gc": mean_gc,
            "duplicate_id_count": len(self.duplicate_ids),
            "invalid_records": self.invalid_records,
        }
def iter_fastq(path: Path) -> Iterator[FastqRecord]:
    opener = gzip.open if path.suffix == ".gz" else open
    with opener(path, "rt", encoding="utf-8") as handle:
        while True:
            header = handle.readline()
            if not header:
                return
sequence = handle.readline().rstrip("\n")
            plus = handle.readline().rstrip("\n")
            quality = handle.readline().rstrip("\n")
if not plus.startswith("+"):
                raise ValueError("Malformed FASTQ record: missing '+' line")
record = FastqRecord(
                identifier=header.rstrip("\n")[1:].split()[0],
                sequence=sequence,
                quality=quality,
            )
            record.validate()
            yield record
def run_qc(path: Path) -> dict:
    report = FastqQC()
    seen_ids: set[str] = set()
for record in iter_fastq(path):
        report.add(record, seen_ids)
return report.summary()
if __name__ == "__main__":
    report = run_qc(Path("data/raw/sample_R1.fastq.gz"))
    print(report)
```

عندما يتعامل الكود مع ملفات أو جداول، تصبح هوية الملف وترميزه ومساره جزءًا من النتيجة القابلة لإعادة الإنتاج. القراءة الناجحة للملف لا تعني أن محتواه صالح بيولوجيًا؛ لذلك يجب فصل فحص البنية عن تفسير البيانات. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

The important lesson is not merely that a class can be defined. FastqRecord owns the state and validation rules of one read; FastqQC owns accumulated sample-level

state; run_qc connects the layers without duplicating their internal logic. This separation makes testing, reuse, and replacement of individual components easier.

هذا المثال يوضح لماذا لا ينبغي اختزال OOP في تعريف class ثم استدعاء method واحدة. FastqRecord يحمل البيانات التي تنتمي إلى قراءة واحدة، ويضع قواعدها بالقرب من البيانات نفسها. FastqQC يجمع الحالة المتراكمة بدل تمرير عشرات المتغيرات بين الدوال. أما run_qc فيربط طبقات البرنامج دون أن يعرف تفاصيل كل عملية داخل الكائنات. هذه القسمة تجعل الاختبار، وإعادة الاستخدام، واستبدال جزء من التنفيذ أسهل.

Testing success and failure cases

اختبار حالات النجاح والفشل

```
def test_fastq_record_validation():
    record = FastqRecord("read1", "ATGC", "IIII")
    record.validate()
    assert record.length == 4
    assert record.gc_fraction == 0.5
    assert record.mean_quality() == 40.0
def test_rejects_length_mismatch():
    record = FastqRecord("read2", "ATGC", "III")
    try:
        record.validate()
    except ValueError as exc:
        assert "lengths differ" in str(exc)
    else:
        raise AssertionError("Expected ValueError")
def test_rejects_invalid_base():
    record = FastqRecord("read3", "ATGX", "IIII")
    try:
        record.validate()
    except ValueError as exc:
        assert "Unexpected bases" in str(exc)
    else:
        raise AssertionError("Expected ValueError")
if __name__ == "__main__":
    test_fastq_record_validation()
    test_rejects_length_mismatch()
    test_rejects_invalid_base()
    print("All FastqRecord tests passed.")
```

الدوال المعرفة هنا (test_fastq_record_validation, test_rejects_length_mismatch, test_rejects_invalid_base) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها.

Tests for both valid and invalid states are essential. A utility may appear correct on a tiny file and still fail when sequence and quality lengths differ or an unexpected

symbol appears. These tests validate the class itself before it becomes part of a larger RNA-seq workflow.

وجود اختبارات لحالة صحيحة وحالات فشل ليس ترفًا. الكود قد يعطي ناتجًا يبدو صحيحًا على ملف صغير بينما يفشل عند اختلاف طول الجودة عن التسلسل أو ظهور رمز غير متوقع. الاختبار هنا يثبت سلوك الفئة نفسها، ولذلك يمكن تشغيله قبل إدخالها في خط RNA-seq أكبر.

Part 4 — The Practical Core: Biopython

الباب 4 — القلب النابض: Biopython

الفكرة المركزية: تمثيل التسلسل، الاتجاه، translation، reverse complement وحدود التفسير.

Chapter 10 — Seq: Sequence as an Object

الفصل 10 — Seq: التسلسل ككائن

شرح أعمق للمفهوم

Deeper Conceptual Explanation

التعامل مع Seq يوضح الفرق بين تمثيل السلسلة وتنفيذ قاعدة بيولوجية عليها. reverse_complement لا يعكس النص فقط، بل يطبق التكامل القاعدي ثم يعكس الاتجاه. وtranslate يطبق جدول الشفرة الوراثية وفق إطار القراءة المختار. لذلك يجب أن يكون معنى كل عملية مفهوماً قبل قراءة مخرجاتها.

Working with Seq exposes the difference between representing a string and applying a biological rule to it. reverse_complement does not merely reverse text; it complements bases and then reverses orientation. translate applies the genetic code according to a chosen reading frame. The meaning of each operation should therefore be understood before interpreting its output.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Plain String vs. Seq Object

1. النص العادي مقابل كائن Seq

In Python, a plain string is just a sequence of characters. the workflow can reverse it, slice it, or search it, but it carries no knowledge of what these characters represent. The string "ATGC" is no different from "ABCD" from Python's perspective. A Seq object in Biopython, however, takes the same letters but knows

they represent a DNA sequence, knows the rules of complementarity (A with T, C with G), knows orientation (5' to 3'), and knows how to translate to protein.

في Python، النص العادي هو مجرد سلسلة من الحروف. يمكن عكسه، أو تقطيعه، أو البحث فيه، لكنه لا يحمل أي معرفة بما تمثله هذه الحروف. النص "ATGC" لا يختلف عن النص "ABCD" من وجهة نظر Python. أما كائن Seq في Biopython، فهو يأخذ نفس الحروف لكنه يعرف أنها تمثل تسلسلاً DNA، ويعرف قواعد التكامل (A مع T، و C مع G)، ويعرف الاتجاهية (5' إلى 3')، ويعرف كيف يترجم إلى بروتين.

```
from Bio.Seq import Seq
# نص عادي
text = "ATGC"
print(text[::-1])  # عكس النص فقط: CGTA
# كائن Seq
dna = Seq("ATGC")
print(dna.reverse_complement())  # المتمم العكسي البيولوجي: GCAT
```

This simple example illustrates the fundamental difference: a plain string simply reverses the characters, while Seq applies the biological complementarity rule and then reverses the orientation. The result is not just reversed letters; it is an accurate representation of what happens in nature when we look at the complementary strand.

هذا المثال البسيط يوضح الفرق الجوهري: النص العادي يعكس الحروف فقط، أما Seq فيطبق قاعدة التكامل البيولوجي ثم يعكس الاتجاه. النتيجة ليست مجرد حروف معكوسة؛ إنها تمثيل دقيق لما يحدث في الطبيعة عندما ننظر إلى الشريط المكمل.

2. Orientation

2. الاتجاهية (Orientation)

In Biopython, a Seq object implicitly carries the 5' to 3' orientation. This is important because many biological operations depend on orientation. When computing the reverse complement, Biopython first computes the complement (A→T, T→A, C→G, G→C) and then reverses the order, simulating the complementary strand which is read from 5' to 3' in the opposite direction. Understanding this orientation is essential for correctly interpreting Biopython's results.

في Biopython، كائن Seq يحمل ضمنياً اتجاه 5' إلى 3'. هذا مهم لأن العديد من العمليات البيولوجية تعتمد على الاتجاه. عند حساب المتمم العكسي، يقوم Biopython أولاً بحساب المتمم (A→T، T→A، C→G، G→C) ثم يعكس الترتيب، محاكياً بذلك الشريط المكمل الذي يُقرأ من 5' إلى 3' في الاتجاه المعاكس. فهم هذه الاتجاهية هو أساسي لتفسير نتائج Biopython بشكل صحيح.

```
dna = Seq("ATGC")
print(dna.complement())         # TACG (المتمم)
print(dna.reverse_complement())  # GCAT (المتمم العكسي)
```

Notice that complement() only substitutes letters without changing order, while reverse_complement() both substitutes and reverses. This reflects the difference between the parallel complementary strand and the antiparallel complementary strand in double-stranded DNA.

لاحظ أن complement() فقط يستبدل الحروف دون تغيير الترتيب، بينما reverse_complement() يستبدل ويعكس. هذا يعكس الفرق بين الشريط المكمل الموازي والشريط المكمل المضاد للتوازي في الـ DNA المزدوج.

3. الترجمة (Translation)

3. Translation

One of the most powerful features of the Seq object is translation to protein. Using translate(), the workflow can convert a DNA sequence into a string of amino acids according to the genetic code. Biopython uses the standard genetic table by default, but the workflow can choose other tables depending on the organism. Translation is a purely computational operation: it reads the sequence three bases at a time (codon), and looks up the corresponding amino acid. But it is not evidence that this protein actually exists or performs a function.

واحدة من أقوى ميزات كائن Seq هي الترجمة إلى بروتين. باستخدام translate()، يمكن تحويل تسلسل DNA إلى سلسلة من الأحماض الأمينية وفقاً للشفرة الوراثية. Biopython يستخدم الجدول الوراثي القياسي افتراضياً، لكن يمكن اختيار جداول أخرى حسب الكائن الحي. الترجمة هي عملية حسابية بحتة: تقرأ التسلسل ثلاث قواعد في كل مرة (كودون)، وتبحث عن الحمض الأميني المقابل. لكنها ليست دليلاً على أن هذا البروتين موجود فعلياً أو يؤدي وظيفة.

Purpose: apply the genetic code to a defined reading frame; the output is computational and requires context for biological interpretation.

الغرض: تطبيق جدول الشفرة الوراثية على إطار قراءة محدد؛ الناتج حسابي ويحتاج إلى سياق عند تفسيره بيولوجياً.

```
dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
protein = dna.translate()
print(protein)  # سلسلة الأحماض الأمينية المترجمة
```

This example illustrates translation, but the real question is: does this protein exist in nature? Does it have a function? Biopython does not answer this question. Translation is a computational tool for generating hypotheses, not a means of proving biological facts.

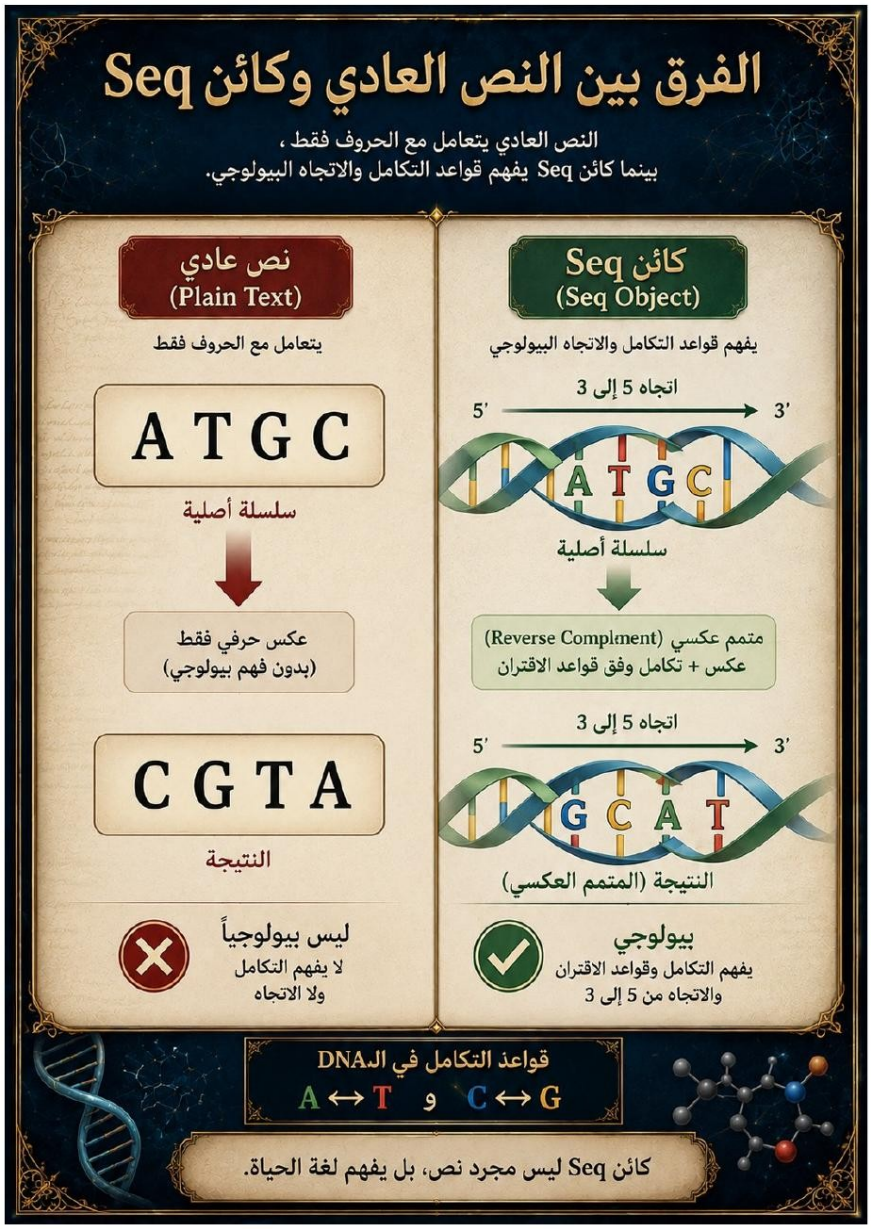
هذا المثال يوضح الترجمة، لكن السؤال الحقيقي هو: هل هذا البروتين موجود في الطبيعة؟ هل له وظيفة؟ Biopython لا يجيب عن هذا السؤال. الترجمة هي أداة حسابية لتوليد فرضيات، وليست وسيلة لإثبات الحقائق البيولوجية.

4. Limits of Interpretation: The Computational Operation Is Not Biological Proof

4. حدود التفسير: العملية الحسابية ليست دليلاً بيولوجياً

This is the most important lesson in this chapter. Biopython performs precise computational operations: the reverse complement is a mathematical transformation on a string of letters, and translation is a table lookup. These operations are computationally correct, but they prove nothing about the biological world. When you translate a sequence and obtain a protein, you get a string of amino acids that might represent a protein, but proving that this protein exists, or that it performs a function, requires independent evidence: laboratory experiments, database comparisons, or functional studies.

هذا هو أهم درس في هذا الفصل. Biopython ينفذ عمليات حسابية محددة بدقة: المتمم العكسي هو تحويل رياضي على سلسلة من الحروف، والترجمة هي بحث في جدول. هذه العمليات صحيحة حسابياً، لكنها لا تثبت أي شيء عن العالم البيولوجي. عندما تترجم تسلسلاً وتحصل على بروتين، فأنت تحصل على سلسلة من الأحماض الأمينية التي قد تمثل بروتيناً، لكن إثبات أن هذا البروتين موجود، أو أنه يؤدي وظيفة، يحتاج إلى أدلة مستقلة: تجارب معملية، أو مقارنات بقواعد البيانات، أو دراسات وظيفية.



Infographic 11 / إنفوجرافيك 11


```
# حسا بي  بحت
dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
protein = dna.translate()
```

print(protein) # هذا مخرجات حسابية، وليست حقيقة بيولوجية

```
# السؤال البيولوجي: هل هذا البروتين موجود في الخلية؟
# Biopython لا يجيب. تحتاج إلى أدلة إضافية.
```

Confusing these two levels — the computational and the biological — is one of the most common mistakes in bioinformatics. This chapter clearly draws the line between them.

5. Practical Application

الخلط بين هذين المستويين — الحسابي والبيولوجي — هو أحد أكثر الأخطاء شيوعاً في المعلوماتية الحيوية. هذا الفصل يضع الخط الفاصل بينهما بوضوح.

5. التطبيق العملي

Let us apply what we have learned in a simple exercise. the workflow has a DNA sequence, and you want to calculate its length, GC content, reverse complement, and translate it to protein. Using Seq:

دعنا نطبق ما تعلمناه في تمرين بسيط. يوجد في المثال تسلسل DNA، وتريد حساب طوله، ومحتواه GC، ومتممه العكسي، وترجمته إلى بروتين. باستخدام Seq:

from Bio.Seq import Seq
from Bio.SeqUtils import gc_fraction

تسلسل اصطناعي

dna_seq = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC")

العمليات الأساسية

```
print(f"التسلسل: {dna_seq}")
print(f"الطول: {len(dna_seq)}")
print(f"نسبة GC: {gc_fraction(dna_seq)*100:.2f}%")
print(f"المتمم العكسي: {dna_seq.reverse_complement()}")
print(f"الترجمة (إطار 1): {dna_seq.translate()}")
# الترجمة في أطر قراءة مختلفة
print(f"الترجمة (إطار 2): {dna_seq[1:].translate()}")
print(f"الترجمة (إطار 3): {dna_seq[2:].translate()}")
```

Note that GC() is a function from Bio.SeqUtils for calculating GC content, and is not part of the Seq object itself. This shows that some operations are helper functions, while fundamental operations are methods of the object.

لاحظ أن GC() هي دالة من Bio.SeqUtils لحساب نسبة GC، وليست جزءاً من كائن Seq نفسه. هذا يوضح أن بعض العمليات هي دوال مساعدة، بينما العمليات الأساسية هي طرق داخل الكائن.

Chapter Summary

خلاصة الفصل

The Seq object is the heart of Biopython. It is more than just text; it is an intelligent representation of a biological sequence that knows orientation, complementarity rules, and how to translate. But its power comes with responsibility: the operations it provides are purely computational, and interpreting them biologically requires context and independent evidence. Distinguishing between what Seq does (computational operations) and what we infer from it (biological inference) is the essence of critical thinking in bioinformatics. The next chapter will take you to the next level: SeqRecord, where the sequence gains its identity and metadata.

كائن Seq هو جوهر Biopython. هو أكثر من مجرد نص؛ هو تمثيل ذكي للتسلسل البيولوجي يعرف الاتجاهية، وقواعد التكامل، وكيفية الترجمة. لكن قوته تأتي مع مسؤولية: العمليات التي يوفرها هي حسابية بحتة، وتفسيرها بيولوجياً يتطلب سياقاً وأدلة مستقلة. التمييز بين ما يفعله Seq (عمليات حسابية) وما نستنتجه منه (استدلال بيولوجي) هو جوهر التفكير النقدي في المعلوماتية الحيوية. الفصل القادم سيأخذك إلى المستوى التالي: SeqRecord، حيث يضاف إلى التسلسل هويته وبياناته الوصفية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Seq: Sequence as an Object" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Seq: التسلسل ككائن" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "the Seq object represents a real DNA molecule with all its biological properties." This is incorrect; the Seq object represents the sequence as text with computational operations, but it does not represent the three-dimensional structure or cellular context. The correct statement is: "the Seq object provides a computational representation of the sequence with biologically meaningful operations (such as complementation and translation), but it is not a model of the living molecule." A simple test: Practical check: "Does this operation (e.g., translation) tell me about a real function in the cell, or is it just a computational transformation that needs confirmation?".

مثال على التجاوز: القول إن "كائن Seq يمثل جزيء DNA حقيقياً بكل خصائصه البيولوجية". هذا غير صحيح؛ كائن Seq يمثل التسلسل كنص مع عمليات حسابية، لكنه لا يمثل البنية ثلاثية الأبعاد أو السياق الخلوي. الصحيح هو: "كائن Seq يوفر تمثيلاً حسابياً للتسلسل مع عمليات ذات معنى بيولوجي (مثل التكامل والترجمة)، لكنه ليس نموذجاً

للجزيء الحي." اختبار بسيط: فحص تطبيقي: "هل هذه العملية (مثل الترجمة) تخبرني عن وظيفة حقيقية في الخلية، أم أنها مجرد تحويل حسابي يحتاج إلى تأكيد؟"ـ

Example: Computing the reverse complement using dna.reverse_complement() is a purely computational representation. Claiming that "this complementary sequence is the second strand of DNA in the cell" is a biological claim not justified by the computational operation alone. The reverse complement is a mathematical transformation on letters, not a discovery of a real DNA strand. Practical test: before using any Seq result as biological evidence, verify that there is context (such as cell type, experimental conditions) and independent evidence supporting the project's conclusion.

مثال: حساب المتمم العكسي باستخدام dna.reverse_complement() هو تمثيل حسابي بحت. الادعاء بأن "هذا التسلسل المتمم هو الشريط الثاني للـ DNA في الخلية" هو ادعاء بيولوجي غير مبرر بالعملية الحسابية وحدها. المتمم العكسي هو تحويل رياضي على الحروف، وليس اكتشافاً لشريط DNA حقيقي. اختبار عملي: قبل أن تستخدم أي نتيجة من Seq كدليل بيولوجي، تحقق من وجود سياق (مثل نوع الخلية، الظروف التجريبية) وأدلة مستقلة تدعم استنتاجك.

مثال تطبيقي على دالة translate() في Biopython:

Applied example of the translate() function in Biopython:

عند استخدام Seq وطرقه، يجب تسجيل ما يلي:

· المصدر: التسلسل المستخدم (رقم الانضمام، الكائن الحي، النسيج، طريقة الحصول عليه)ـ

· المعلمات: جدول الشفرة الوراثية (إذا كان مختلفاً عن الافتراضي)، وأي معاملات أخرى استخدمتها (مثل to_stop=True في الترجمة)ـ

· التحويلات: أي خطوات قمت بها قبل استخدام Seq (مثل قص التسلسل، أو تغيير الحالة إلى أحرف كبيرة)ـ

When using Seq and its methods, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة نفس العمليات على نفس التسلسل باستخدام نفس الإصدارات والمعلمات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the sequence used (accession number, organism, tissue, how it was obtained).

· Parameters: the genetic code table (if different from default), and any other parameters you used (e.g., to_stop=True in translation).

· Transformations: any steps you performed before using Seq (e.g., trimming the sequence, or converting to uppercase).

Test: Practical check: "If I gave this record to a colleague a year later, could they reproduce the same operations on the same sequence using the same versions and parameters?" If the answer is no, the documentation is insufficient.

قبل أن تستخدم Seq على تسلسل حقيقي من مريض أو من قاعدة بيانات، اختبر طرق Seq على تسلسل اصطناعي صغير معروف النتيجة. استخدم التسلسل "ATGC" وتأكد أن reverse_complement() يعطي "GCAT". استخدم التسلسل "ATGGCC" (الذي يترجم إلى "MA") وتأكد أن translate() يعطي النتيجة المتوقعة. استخدم التسلسل "TAA" (كودون توقف) وتأكد أن translate() يتعامل معه بشكل صحيح. إذا نجحت هذه الاختبارات، فأنت واثق من أن Seq يعمل كما هو متوقع، وأي نتيجة تحصل عليها من بيانات حقيقية تعكس خصائص البيانات وليس خطأ في استخدامك للمكتبة.

Before using Seq on a real sequence from a patient or a database, test Seq's methods on a small synthetic sequence with a known outcome. Use the sequence "ATGC" and verify that reverse_complement() returns "GCAT". Use the sequence "ATGGCC" (which translates to "MA") and verify that translate() gives the expected result. Use the sequence "TAA" (a stop codon) and verify that translate() handles it correctly. If these tests succeed, the workflow is confident that Seq works as expected, and any result you get from real data reflects the properties of the data, not an error in the project's use of the library.

Chapter 11 — SeqRecord: Sequence with Identity

الفصل 11 — SeqRecord: التسلسل مع هويته

الفكرة المركزية: المعرف، الوصف، provenance، annotations، التتبع.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

التسلسل بلا هوية قابل للضياع داخل مشروع كبير. SeqRecord يربط الحروف بمعرف ووصف وتعليقات وبيانات وصفية، وبذلك يصبح السجل قابلاً للتتبع. فقدان هذه الطبقة أثناء تحويل الملفات قد لا يغير الحروف، لكنه يغير قيمة السجل العلمي وقدرته على التفسير وإعادة الاستخدام.

A sequence without identity can easily become detached from its meaning in a large project. SeqRecord connects the letters to an identifier, description, annotations, and metadata, making the record traceable. Losing this layer during

file conversion may leave the letters unchanged while reducing the scientific value and reusability of the record.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. ما هو SeqRecord؟

1. What Is SeqRecord؟

SeqRecord is an object in Biopython that combines a sequence (a Seq object) with a set of metadata that gives it scientific context. It is the complete record of the sequence: it contains the sequence itself, a unique identifier (such as a gene name or database accession number), a brief description, a dictionary of metadata (annotations), and a list of features that mark important locations on the sequence (such as gene locations, exons, or translation start sites).

SeqRecord هو كائن في Biopython يجمع بين التسلسل (كائن Seq) ومجموعة من البيانات الوصفية التي تمنحه سياقاً علمياً. هو السجل الكامل للتسلسل: يحتوي على التسلسل نفسه، ومعرف فريد (مثل اسم الجين أو رقم الانضمام في قاعدة البيانات)، ووصف مختصر، وقاموس من البيانات الوصفية (annotations)، وقائمة من الميزات (features) التي تحدد مواقع مهمة على التسلسل (مثل مواقع الجينات، أو الإكسونات، أو مواقع بدء الترجمة).

In practice, when you read a FASTA file using SeqIO.read(), you get a SeqRecord object, not just a Seq. This ensures that the identity and description travel with the sequence wherever they go. Without this record, you might lose the context that gives the sequence its meaning.

في الممارسة العملية، عندما تقرأ ملف FASTA باستخدام SeqIO.read()، تحصل على كائن SeqRecord، وليس مجرد Seq. هذا يضمن أن الهوية والوصف يسافران مع التسلسل أينما ذهب. بدون هذا السجل، قد تفقد السياق الذي يمنح التسلسل معناه.

2. مكونات SeqRecord

2. Components of SeqRecord

The SeqRecord object consists of several main parts:

كائن SeqRecord يتكون من عدة أجزاء رئيسية:

- seq.: كائن Seq الذي يمثل التسلسل نفسه.

· .id: معرف فريد للتسلسل (مثل رقم الانضمام في GenBank أو اسم الجين). يجب أن يكون فريداً في سياق مجموعة البيانات.

· .name: اسم مختصر أو بديل (غالباً ما يكون مشابهاً لـ .id ولكن يمكن أن يكون مختلفاً).

· .description: وصف نصي للتسلسل (مثل اسم الكائن الحي، والجين، والوظيفة المحتملة).

· .seq: The Seq object representing the sequence itself.

· .annotations: قاموس (dictionary) يحتوي على بيانات وصفية إضافية، مثل نوع الجزيء (DNA، RNA، بروتين)، أو تاريخ الإنشاء، أو المرجع.

· .id: A unique identifier for the sequence (e.g., GenBank accession number or gene name). It should be unique within the dataset context.

· .features: قائمة من كائنات SeqFeature التي تمثل ميزات موضعية على التسلسل (مثل مواقع الجينات، الإكسونات، مناطق الربط).

· .name: A short or alternative name (often similar to .id but may differ).

· .description: A textual description of the sequence (e.g., organism name, gene, potential function).

· .annotations: A dictionary containing additional metadata, such as molecule type (DNA, RNA, protein), creation date, or references.

· .features: A list of SeqFeature objects representing localized features on the sequence (e.g., gene locations, exons, binding sites).

```
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
# إنشاء تسلسل
my_seq = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC")
# إنشاء SeqRecord
record = SeqRecord(
    seq=my_seq,
    id="SYN_001",
    name="synthetic_gene",
    description="تسلسل اصطناعي للتمرين على Biopython",
    annotations={"molecule_type": "DNA", "source": "synthetic", "date": "2025-01-15"}
)
```

```
# عرض محتويات السجل
print(f"المعرف: {record.id}")
print(f"الوصف: {record.description}")
print(f"البيانات الوصفية: {record.annotations}")
```

3. The Value of Identity and Provenance

Why is identity so important? Because a sequence without identity loses its scientific value. For illustration, consider a case in which find a sequence in a file, but you do not know which organism, which tissue, or what its potential function is. the workflow can calculate its properties (length, GC content, translation), but you cannot interpret them biologically. Identity is what links the sequence to existing knowledge: databases, scientific literature, and previous experiments.

3. قيمة الهوية والمنشأ (Provenance)

Provenance is the record of the sequence's history: where it came from, how it was produced, what transformations were applied to it. In SeqRecord, the workflow can store part of this provenance in the annotations field (e.g., date, source, sequencing method). Good provenance is what allows others to assess the credibility of the project’s data and reproduce the project’s work.

لماذا الهوية مهمة بهذا الشكل؟ لأن التسلسل بدون هوية يفقد قيمته العلمية. للتوضيح، يمكن تصور حالة يكون فيها تجد تسلسلاً في ملف، لكنك لا تعرف من أي كائن حي، ولا من أي نسيج، ولا ما هي وظيفته المحتملة. يمكن حساب خصائصه (الطول، نسبة GC، الترجمة)، لكن لا يمكن تفسيرها بيولوجياً. الهوية هي ما يربط التسلسل بالمعرفة الموجودة: قواعد البيانات، الأدبيات العلمية، والتجارب السابقة.

المنشأ (Provenance) هو سجل تاريخ التسلسل: من أين جاء؟ كيف أُنتج؟ ما هي التحويلات التي طُبقت عليه؟ في SeqRecord، يمكن تخزين جزء من هذا المنشأ في حقل annotations (مثل التاريخ، المصدر، طريقة التسلسل). المنشأ الجيد هو ما يسمح للآخرين بتقييم مصداقية بياناتك وإعادة إنتاج عملك.

```
# إضافة منشأ إلى السجل
record.annotations["sequencing_technology"] = "Illumina"
record.annotations["lab"] = "Bioinformatics Lab, University X"
record.annotations["sample_id"] = "Tumor_01"
```

4. Features and Their Role

Features (SeqFeature) are Biopython's way of representing additional information about specific locations on the sequence. Instead of writing separate comments, the workflow can add features to the SeqRecord that define the start and end of a specific region, its strand, and its type (gene, exon, coding region, etc.). This makes

the record rich in information and allows programs to automatically understand the sequence's structure.

4. الميزات (Features) ودورها

الميزات (SeqFeature) هي طريقة Biopython لتمثيل المعلومات الإضافية حول مواقع محددة على التسلسل. بدلاً من كتابة تعليقات منفصلة، يمكن إضافة ميزات إلى SeqRecord تحدد بداية ونهاية منطقة معينة، واتجاهها، ونوعها (جين، إكسون، منطقة مشفرة، إلخ). هذا يجعل السجل غنياً بالمعلومات، ويسمح للبرامج بفهم بنية التسلسل تلقائياً.

from Bio.SeqFeature import SeqFeature, FeatureLocation

إنشاء ميزة تمثل جيناً (من القاعدة 5 إلى 50 على الشريط الموجب)

gene_feature = SeqFeature(

```
    FeatureLocation(start=5, end=50, strand=+1),
    type="gene",
    qualifiers={"gene": "BRCA1", "note": "مثال تعليمي لجين وهمي"}
)

# عرض الميزات
for feature in record.features:
    print(f"النوع: {feature.type}, الموقع: {feature.location}, المعرّف: {feature.qualifiers.get('gene')}")
# إضافة الميزة إلى السجل
record.features.append(gene_feature)
```

5. Separating a Sequence from Its Identity: Why Is It a Mistake؟

A common mistake in practice is for a researcher to copy a sequence from one file to another without transferring its identity and metadata. They may keep the letters, but they lose the evidence that allows interpretation. Imagine this sequence represents an important gene associated with a disease. If you lose its identity, you will not know which patient it belongs to, which tissue, or which previous study. The sequence may be computationally correct, but it becomes scientifically meaningless. This chapter emphasizes a golden rule: identity is part of the data, not something separate that can be added later.

5. فصل التسلسل عن هويته: لماذا هو خطأ؟

من الأخطاء الشائعة في الممارسة العملية أن يقوم الباحث بنسخ تسلسل من ملف إلى آخر دون أن ينقل معه الهوية والبيانات الوصفية. قد يحتفظ بالحروف، لكنه يفقد الدليل الذي يسمح بتفسيرها. تخيل أن هذا التسلسل يمثل جيناً مهماً مرتبطاً بمرض. إذا فقدت هويته، فلن تعرف أي مريض ينتمي إليه، ولا أي نسيج، ولا أي دراسة سابقة. قد يكون

التسلسل صحيحاً حسابياً، لكنه يصبح عديم المعنى علمياً. هذا الفصل يؤكد على قاعدة ذهبية: الهوية جزء من البيانات، وليست شيئاً منفصلاً يمكن إضافته لاحقاً.

```
# العمل الخطأ: نسخ التسلسل فقط
seq_only = record.seq # هذا يفقد الهوية
# العمل الصحيح: الاحتفاظ بالسجل كاملاً
# full_record = record  أو نسخ السجل مع جميع بياناته الوصفية
```

6. Practical Application: Reading and Writing Records

When you read a FASTA file using SeqIO, you get a SeqRecord object. When writing, you must provide a SeqRecord object to preserve identity and metadata.

6. التطبيق العملي: قراءة وكتابة السجلات

عندما تقرأ ملف FASTA باستخدام SeqIO، تحصل على كائن SeqRecord. عند الكتابة، يجب أن توفر كائن SeqRecord للحفاظ على الهوية والبيانات الوصفية.

```
from Bio import SeqIO
# قراءة ملف FASTA والحصول على SeqRecord
record = SeqIO.read("example.fasta", "fasta")
print(f"تم قراءة السجل: {record.id} - {record.description}")
# تعديل السجل (مثل إضافة ميزة)
record.annotations["processed"] = "2025-01-15"
# كتابة السجل إلى ملف جديد
SeqIO.write(record, "example_modified.fasta", "fasta")
```

Note that the FASTA format does not carry all the information in a SeqRecord (such as features and complex metadata). This is a limitation of the format, not of Biopython. If you want to preserve all information, use richer formats like GenBank.

لاحظ أن صيغة FASTA لا تحمل كل المعلومات الموجودة في SeqRecord (مثل الميزات والبيانات الوصفية المعقدة). هذا هو قيد الصيغة، وليس قيد Biopython. إذا كنت تريد الحفاظ على كل المعلومات، استخدم صيغاً أكثر غنى مثل GenBank.

7. حدود SeqRecord

7. Limitations of SeqRecord

SeqRecord is a rich object, but it is not a complete model of the living molecule. It holds the sequence, metadata, and features, but it does not represent three-dimensional structure, molecular interactions, or cellular context. Also, the accuracy of the metadata depends on what the user provides. If you enter an incorrect identifier, SeqRecord will still carry it, but it will be misleading.

Therefore, validating metadata is the researcher's responsibility, not Biopython's task.

SeqRecord هو كائن غني، لكنه ليس نموذجاً كاملاً للجزيء الحي. يحتفظ بالتسلسل والبيانات الوصفية والميزات، لكنه لا يمثل البنية ثلاثية الأبعاد، أو التفاعلات الجزيئية، أو السياق الخلوي. كما أن صحة البيانات الوصفية تعتمد على ما يقدمه المستخدم. إذا أدخلت معرفاً خاطئاً، فسيظل SeqRecord يحمله، لكنه سيصبح مضللاً. لذلك، التحقق من صحة البيانات الوصفية هو مسؤولية الباحث، وليست مهمة Biopython.

Chapter Summary

خلاصة الفصل

SeqRecord is the object that transforms a sequence from mere letters into a complete scientific record. It combines the sequence (a Seq object) with identity (id), description, metadata (annotations), and features (features). Identity and provenance are not decoration; they are part of the data, separating a sequence that is interpretable from one that is meaningless. Reading FASTA files using SeqIO returns a SeqRecord, not just a Seq, constantly reminding you that the sequence in its scientific context is more important than the sequence alone. The next chapter will take you a step further, to represent localized features using SeqFeature.

SeqRecord هو الكائن الذي يحول التسلسل من مجرد حروف إلى سجل علمي كامل. يجمع بين التسلسل (كائن Seq) والهوية (id)، والوصف، والبيانات الوصفية (annotations)، والميزات (features). الهوية والمنشأ ليسا مجرد زينة؛ هما جزء من البيانات، ويفصلان التسلسل القابل للتفسير عن التسلسل عديم المعنى. قراءة ملفات FASTA باستخدام SeqIO تعيد لك SeqRecord، وليس مجرد Seq، وهذا يذكرك دائماً بأن التسلسل في سياقه العلمي أهم من التسلسل وحده. الفصل القادم سيأخذك خطوة أبعد، لتمثيل الميزات الموضعية باستخدام SeqFeature.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "SeqRecord: Sequence with Identity" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "SeqRecord: التسلسل مع هويته" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "SeqRecord provides all the information needed to understand the molecule biologically." This is incorrect; SeqRecord provides context (identity, description, features), but it does not provide three-dimensional structure or molecular interactions. The correct statement is:

"SeqRecord wraps the sequence with metadata that gives it scientific context, making it traceable and comparable, but understanding the molecule biologically requires additional evidence." A simple test: Practical check: "If I have this SeqRecord, do I know everything about this molecule in the cell?" If the answer is yes, the workflow is overstepping the limits of what the record provides.

مثال على التجاوز: القول إن "SeqRecord يوفر كل المعلومات اللازمة لفهم الجزيء بيولوجياً". هذا غير صحيح؛ SeqRecord يوفر السياق (الهوية، الوصف، الميزات)، لكنه لا يوفر البنية ثلاثية الأبعاد أو التفاعلات الجزيئية. الصحيح هو: "SeqRecord يغلف التسلسل بالبيانات الوصفية التي تمنحه سياقاً علمياً، مما يجعله قابلاً للتتبع والمقارنة، لكن فهم الجزيء بيولوجياً يحتاج إلى أدلة إضافية." اختبار بسيط: فحص تطبيقي: "إذا كان لدي هذا SeqRecord، هل أعرف كل شيء عن هذا الجزيء في الخلية؟" إذا كانت الإجابة نعم، فأنت تتجاوز حدود ما يقدمه السجل.

Example: Storing a sequence in a SeqRecord with the identifier "BRCA1" is a computational representation. Claiming that "this sequence represents the human BRCA1 gene that causes cancer" is a biological claim not proven by the record. The record tells us that the user claimed it is BRCA1, but it does not prove it. Proving identity and function requires independent evidence (e.g., database matching, laboratory experiments). Practical test: before relying on an identity in a SeqRecord, verify its source and confirm it with external references.

مثال: تخزين تسلسل في SeqRecord مع معرف "BRCA1" هو تمثيل حسابي. الادعاء بأن "هذا التسلسل يمثل جين BRCA1 البشري الذي يسبب السرطان" هو ادعاء بيولوجي لا يثبته السجل. السجل يخبرنا أن المستخدم ادعى أن هذا هو BRCA1، لكنه لا يثبت ذلك. إثبات الهوية والوظيفة يحتاج إلى أدلة مستقلة (مثل مطابقة مع قاعدة بيانات، أو تجارب معملية). اختبار عملي: قبل أن تعتمد على هوية في SeqRecord، تحقق من مصدرها وتأكد من صحتها بمصادر خارجية.

مثال تطبيقي على SeqRecord:

· المدخلات: كائن Seq، معرف فريد، وصف، قاموس annotations (اختياري)، قائمة features (اختياري).

· الافتراضات: أن المعرف فريد في سياق مجموعة البيانات، وأن البيانات الوصفية صحيحة وكاملة.

· المخرجات: كائن SeqRecord يغلف التسلسل بسياقه العلمي.

· القيود: SeqRecord لا يتحقق من صحة المعرف، ولا يتحقق من أن التسلسل يتوافق مع المعرف (مثل أن يكون تسلسل BRCA1 فعلاً)، ولا يوفر البنية الجزيئية.

Applied example of SeqRecord:

اختبار: قبل استخدام SeqRecord، تحقق من أن المعرف فريد وأن البيانات الوصفية متسقة مع مصدر التسلسل.

· Inputs: a Seq object, a unique identifier, a description, an annotations dictionary (optional), a list of features (optional).

· Assumptions: that the identifier is unique within the dataset context, and that the metadata is correct and complete.

· Outputs: a SeqRecord object that wraps the sequence with its scientific context.

· Limitations: SeqRecord does not validate the identifier, nor does it verify that the sequence matches the identifier (e.g., that it is actually the BRCA1 sequence), nor does it provide molecular structure.

Test: before using a SeqRecord, verify that the identifier is unique and that the metadata is consistent with the sequence's source.

عند إنشاء أو تعديل SeqRecord، يجب تسجيل ما يلي:

· المصدر: من أين جاء التسلسل (رقم الانضمام، الكائن الحي، النسيج، طريقة التسلسل).

· الإصدار: إصدار Biopython، وإصدار قاعدة البيانات المرجعية إذا كانت مستخدمة.

· المعلمات: أي اختيارات في تصميم السجل (مثل كيفية تنسيق المعرف، أو تنظيم annotations).

· التحويلات: أي خطوات قمت بها قبل إنشاء السجل (مثل قص التسلسل، أو إزالة البادئات).

When creating or modifying a SeqRecord, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا SeqRecord لزميل بعد عام، هل يستطيع فهم من أين جاء التسلسل، وما هي تحويلاته، ويعيد استخدامه في تحليل جديد بثقة؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: where the sequence came from (accession number, organism, tissue, sequencing method).

· Version: Biopython version, and reference database version if used.

· Parameters: any choices in record design (e.g., how to format the identifier, or organize annotations).

· Transformations: any steps you performed before creating the record (e.g., trimming the sequence, or removing primers).

Test: Practical check: "If I gave this SeqRecord to a colleague a year later, could they understand where the sequence came from, what transformations were applied, and reuse it in a new analysis with confidence?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على SeqRecord في تحليل بيانات حقيقية، أنشئ SeqRecord اصطناعياً ببيانات وهمية. استخدم معرفاً وهمياً مثل "TEST_001"، وتسلسلاً صغيراً مثل "ATGC"، وقاموس annotations يحتوي على حقول وهمية. اكتب هذا السجل إلى ملف FASTA أو GenBank، ثم اقرأه مرة أخرى. تحقق من أن جميع الحقول (id، description، annotations) بقيت كما هي عند القراءة. إذا نجحت، فأنت واثق من أن آلية العمل مع SeqRecord سليمة، وأي مشكلة مع بيانات حقيقية تعكس جودة البيانات وليس خطأ في استخدام Biopython.

Before relying on SeqRecord in real data analysis, create a synthetic SeqRecord with dummy data. Use a dummy identifier like "TEST_001", a small sequence like "ATGC", and an annotations dictionary with dummy fields. Write this record to a FASTA or GenBank file, then read it back. Verify that all fields (id, description, annotations) remain as they were when read. If this succeeds, the workflow is confident that the project's workflow with SeqRecord is sound, and any problem with real data reflects data quality, not an error in using Biopython.

Chapter 12 — SeqFeature: Representing Locations and Features

الفصل 12 — SeqFeature: تمثيل المواقع والميزات

الفكرة المركزية: coordinates، strand، feature types، qualifiers، غموض الحدود.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

SeqFeature يصف موضعاً أو ميزة داخل التسلسل؛ لا يجعل الوصف حقيقة بيولوجية بمجرد إضافته إلى الكائن. الإحداثيات والاتجاه ونوع الميزة كلها تمثيلات يجب أن ترتبط بمصدر أو قاعدة تفسيرية مناسبة. هذا الفصل بين التمثيل والدليل مهم خصوصاً عندما تصبح الميزات أساساً لخطوات لاحقة.

SeqFeature describes a location or feature within a sequence; adding it to an object does not make the annotation biologically true. Coordinates, strand, and feature type are representations that should be tied to an appropriate source or interpretive rule. This separation between representation and evidence becomes especially important when annotations feed later analyses.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. ما هو SeqFeature؟

1. What Is SeqFeature؟

SeqFeature is an object in Biopython that represents a specific region on a sequence (such as a gene, exon, or translation start site) with information about its location, strand, type, and additional properties. It is Biopython's way of translating "this part of the sequence is important for this reason" into a language that computers understand.

SeqFeature هو كائن في Biopython يمثل منطقة محددة على التسلسل (مثل جين، أو إكسون، أو موقع بدء الترجمة) مع معلومات عن موقعها واتجاهها ونوعها وخصائصها الإضافية. هو طريقة Biopython لترجمة "هذا الجزء من التسلسل مهم لهذا السبب" إلى لغة يفهمها الحاسوب.

In practice, when you read a GenBank file using SeqIO, you get a SeqRecord containing a list of SeqFeature objects that represent all the features documented in the file. These features are what make a GenBank file rich in information compared to a simple FASTA file.

في الممارسة العملية، عندما تقرأ ملف GenBank باستخدام SeqIO، تحصل على SeqRecord يحتوي على قائمة من SeqFeature التي تمثل جميع الميزات الموثقة في الملف. هذه الميزات هي ما يجعل ملف GenBank غنياً بالمعلومات مقارنة بملف FASTA البسيط.

```
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio.SeqFeature import SeqFeature, FeatureLocation
# إنشاء تسلسل وسجل
my_seq = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC")
record = SeqRecord(my_seq, id="SYN_001", description="تسلسل اصطناعي")
# إنشاء ميزة تمثل جيناً (من القاعدة 5 إلى 25 على الشريط الموجب)
gene_feature = SeqFeature(
    FeatureLocation(start=5, end=25, strand=+1),
    type="gene",
    qualifiers={"gene": "BRCA1", "note": "جين وهمي للتمرين"}
)
# عرض الميزة
print(f"النوع: {gene_feature.type}")
print(f"الموقع: {gene_feature.location}")
print(f"المعرف: {gene_feature.qualifiers.get('gene')}")
# إضافة الميزة إلى السجل
record.features.append(gene_feature)
```

2. Coordinates and Strand

Any location on a sequence is defined by three essential elements: start, end, and strand. These elements precisely determine which part of the sequence the feature refers to.

2. الإحداثيات (Coordinates) والاتجاه (Strand)

أي موقع على التسلسل يُعرّف بثلاثة عناصر أساسية: البداية (start)، النهاية (end)، والاتجاه (strand). هذه العناصر تحدد بدقة أي جزء من التسلسل تقصده الميزة.

· البداية (start) : موضع أول قاعدة في الميزة. في Biopython، الإحداثيات تبدأ من الصفر (zero-based)، وهذا يعني أن أول قاعدة في التسلسل هي 0، وليس 1. هذا يختلف عن بعض التنسيقات الأخرى (مثل GenBank التي تستخدم إحداثيات تبدأ من 1). هذا الاختلاف مصدر شائع للارتباك ويجب فهمه جيداً.

· Start: The position of the first base in the feature. In Biopython, coordinates are zero-based, meaning the first base of the sequence is 0, not 1. This differs from some other formats (e.g., GenBank which uses 1-based coordinates). This difference is a common source of confusion and must be well understood.

· النهاية (end) : موضع آخر قاعدة في الميزة. في Biopython، النهاية تكون غير مشمولة (exclusive)، أي أن الميزة تشمل القواعد من start إلى end-1. هذا متوافق مع أسلوب Python في التقطيع (slicing).

· End: The position of the last base in the feature. In Biopython, the end is exclusive, meaning the feature includes bases from start to end-1. This is consistent with Python's slicing behavior.

· الاتجاه (strand) : يشير إلى الشريط الذي تقع عليه الميزة. القيمة +1 تعني الشريط الموجب (5' إلى 3')، والقيمة -1 تعني الشريط السالب (3' إلى 5')، والقيمة 0 أو None تعني أن الاتجاه غير معروف أو غير محدد.

· Strand: Indicates which strand the feature lies on. A value of +1 means the plus strand (5' to 3'), -1 means the minus strand (3' to 5'), and 0 or None means the strand is unknown or unspecified.

from Bio.SeqFeature import FeatureLocation

موقع من القاعدة 5 إلى القاعدة 15 (يشمل القواعد 5-14)

loc = FeatureLocation(start=5, end=15, strand=+1)

```
print(f"البداية: {loc.start}, النهاية: {loc.end}, الاتجاه: {loc.strand}")
print(f"الطول: {loc.end - loc.start}") # الطول = 10 قواعد
```

Understanding coordinates and strand is not just a technical detail. If you mis-specify the start or end, the feature may point to the wrong location on the sequence. If you mis-specify the strand, you may interpret the feature backwards

(e.g., reading a gene on the minus strand instead of the plus strand). This type of error can completely change the biological conclusion.

3. Feature Types and Qualifiers

فهم الإحداثيات والاتجاه ليس مجرد تفاصيل فنية. إذا أخطأت في تحديد البداية أو النهاية، فقد تشير الميزة إلى موقع خاطئ على التسلسل. وإذا أخطأت في الاتجاه، فقد تفسر الميزة بشكل معاكس (مثل قراءة جين على الشريط السالب بدلاً من الموجب). هذا النوع من الأخطاء يمكن أن يغير الاستنتاج البيولوجي بالكامل.

SeqFeature has a type field that specifies the feature type (such as "gene", "exon", "CDS", "promoter"). This field helps classify features and understand their potential biological function. type is a free text field, but there are common conventions used in databases like GenBank.

3. أنواع الميزات (Feature Types) والمؤهلات (Qualifiers)

In addition to type, a feature can contain qualifiers, which is a dictionary that stores additional information about the feature. For example, a feature of type "gene" might have a gene qualifier carrying the gene name, and a note qualifier carrying additional notes. Qualifiers are how you add rich details about the feature beyond just its type and location.

SeqFeature يحتوي على حقل type يحدد نوع الميزة (مثل "gene"، "exon"، "CDS"، "promoter"). هذا الحقل يساعد في تصنيف الميزات وفهم وظيفتها البيولوجية المحتملة. type هو نص حر، لكن هناك اصطلاحات شائعة مستخدمة في قواعد البيانات مثل GenBank.

بالإضافة إلى type، يمكن للميزة أن تحتوي على qualifiers، وهو قاموس (dictionary) يخزن معلومات إضافية عن الميزة. على سبيل المثال، ميزة من نوع "gene" قد تحتوي على مؤهل gene يحمل اسم الجين، ومؤهل note يحمل ملاحظات إضافية. المؤهلات هي الطريقة التي تضيف بها تفاصيل غنية عن الميزة تتجاوز مجرد النوع والموقع.

```
# ميزة مع مؤهلات متعددة
feature = SeqFeature(
    FeatureLocation(start=10, end=50, strand=+1),
    type="CDS",
    qualifiers={
        "gene": "BRCA1",
        "product": "breast cancer type 1 susceptibility protein",
        "codon_start": 1,
        "translation": "MQL..."
    }
)
```

```
# الوصول إلى المؤهلات
print(f"الجين: {feature.qualifiers.get('gene')}")
print(f"المنتج: {feature.qualifiers.get('product')}")
```

4. Boundary Ambiguity

Sometimes, the boundaries of a feature are not precisely known. There may be multiple possible exons, or the gene start may not be exactly defined. Biopython handles this using special location objects like BeforePosition, AfterPosition, and UnknownPosition. These allow representation of cases where we do not know the exact boundaries. In an educational context, we use clear locations, but it is important to know that Biopython can also represent ambiguity.

4. غموض الحدود (Boundary Ambiguity)

في بعض الأحيان، لا تكون حدود الميزة معروفة بدقة. قد يكون هناك عدة إكسونات محتملة، أو قد تكون بداية الجين غير محددة تماماً. Biopython يتعامل مع هذا باستخدام كائنات مواقع خاصة مثل BeforePosition و AfterPosition و UnknownPosition. هذه تسمح بتمثيل الحالات التي لا نعرف فيها الحدود الدقيقة. في السياق التعليمي، نستخدم مواقع واضحة، لكن من المهم معرفة أن Biopython يمكنه تمثيل الغموض أيضاً.

from Bio.SeqFeature import BeforePosition, AfterPosition

موقع يبدأ قبل القاعدة 5 (غير معروف بدقة)

loc_before = FeatureLocation(start=BeforePosition(5), end=10, strand=+1)

```
print(loc_before)
print(loc_after)
# موقع ينتهي بعد القاعدة 20 (غير معروف بدقة)
loc_after = FeatureLocation(start=5, end=AfterPosition(20), strand=+1)
```

5. Practical Application: Adding Features to a SeqRecord

In a practical application, you might want to add features to a SeqRecord to document important locations. This is especially useful when writing GenBank files, where you want the file to carry all the information needed to interpret the sequence.

5. التطبيق العملي: إضافة ميزات إلى SeqRecord

في تطبيق عملي، قد تريد إضافة ميزات إلى SeqRecord لتوثيق المواقع المهمة. هذا مفيد بشكل خاص عند كتابة ملفات GenBank، حيث تريد أن يحمل الملف كل المعلومات اللازمة لتفسير التسلسل.

from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio.SeqFeature import SeqFeature, FeatureLocation

إنشاء تسلسل وسجل

my_seq = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC" * 10)
record = SeqRecord(my_seq, id="SYN_001", description="جين اصطناعي مع ميزات")

إضافة ميزة جين

gene_feature = SeqFeature(

```
    FeatureLocation(start=0, end=300, strand=+1),
    type="gene",
    qualifiers={"gene": "SYN_GENE_001"}
)
record.features.append(gene_feature)
# إضافة ميزة CDS (منطقة مشفرة)
cds_feature = SeqFeature(
    FeatureLocation(start=10, end=280, strand=+1),
    type="CDS",
    qualifiers={
        "gene": "SYN_GENE_001",
        "product": "hypothetical protein",
        "codon_start": 1
    }
)
record.features.append(cds_feature)
# عرض جميع الميزات
for feature in record.features:
    print(f"{feature.type}: {feature.location} - {feature.qualifiers}")
```

6. Why Distinguish Between Information Representation and Its Practical Use؟

The chapter emphasizes that SeqFeature is a representation of information, not a practical design. Adding a feature to a SeqRecord does not mean that this feature is biologically correct, or that it represents a real gene, or that it is ready for use in gene editing. It is merely a document saying "according to the available information, this location represents X." Moving from this representation to practical use (e.g., designing molecules for gene editing) requires additional steps and independent verification. This distinction is essential to prevent misunderstandings that could lead to incorrect conclusions or unsafe applications.

6. لماذا التمييز بين تمثيل المعلومة واستخدامها العملي؟

الفصل يؤكد على أن SeqFeature هو تمثيل للمعلومة، وليس تصميماً عملياً. إضافة ميزة إلى SeqRecord لا تعني أن هذه الميزة صحيحة بيولوجياً، أو أنها تمثل جيناً حقيقياً، أو أنها جاهزة للاستخدام في تحرير الجينات. هي مجرد وثيقة تقول "وفقاً للمعلومات المتوفرة، هذا الموقع يمثل كذا". الانتقال من هذا التمثيل إلى الاستخدام العملي (مثل

تصميم جزيئات لتعديل جيني) يتطلب خطوات إضافية وتحققاً مستقلاً. هذا التمييز ضروري لمنع سوء الفهم الذي قد يؤدي إلى استنتاجات خاطئة أو تطبيقات غير آمنة.

Chapter Summary

خلاصة الفصل

SeqFeature is Biopython's tool for representing locations and features on a sequence. It uses coordinates (start and end), strand (plus or minus), type, and qualifiers to document the region and its biological significance. Accurate understanding of coordinates and strand is essential, because errors in them change the meaning of the feature. Features make the record rich in information, but they remain a representation of information, not proof of biological truth. The next chapter will tie all these concepts together through SeqIO, the file reading and writing tool.

SeqFeature هو أداة Biopython لتمثيل المواقع والميزات على التسلسل. يستخدم الإحداثيات (بداية ونهاية)، والاتجاه (شريط موجب أو سالب)، والنوع، والمؤهلات لتوثيق المنطقة وأهميتها البيولوجية. فهم الإحداثيات والاتجاه بدقة هو أساسي، لأن الأخطاء فيهما تغير معنى الميزة. الميزات تجعل السجل غنياً بالمعلومات، لكنها تبقى تمثيلاً للمعلومة وليس دليلاً على الحقيقة البيولوجية. الفصل القادم سيربط كل هذه المفاهيم معاً من خلال SeqIO، أداة قراءة وكتابة الملفات.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "SeqFeature: Representing Locations and Features" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "SeqFeature: تمثيل المواقع والميزات" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "the presence of a feature in a SeqRecord proves the existence of a real gene in the sequence." This is incorrect; the feature is a representation of the location according to available information, but it does not prove that this location is a real gene. The correct statement is: "SeqFeature represents a documented location on the sequence with information about its type and properties, but it requires independent evidence (such as database matching or experiments) to confirm its biological identity." A simple test: Practical check: "If this feature is in the record, does that mean the gene actually exists in the cell?" If the answer is yes, the workflow is overstepping the limits of what the representation provides.

مثال على التجاوز: القول إن "وجود ميزة في SeqRecord يثبت وجود جين حقيقي في التسلسل". هذا غير صحيح؛ الميزة هي تمثيل للموقع وفقاً للمعلومات المتوفرة، لكنها لا تثبت أن هذا الموقع هو جين حقيقي. الصحيح هو: "SeqFeature يمثل موقعاً موثقاً على التسلسل مع معلومات عن نوعه وخصائصه، لكنه يحتاج إلى أدلة مستقلة (مثل مطابقة مع قاعدة بيانات أو تجارب) لتأكيد هويته البيولوجية." اختبار بسيط: فحص تطبيقي: "إذا كانت هذه الميزة موجودة في السجل، فهل هذا يعني أن الجين موجود فعلاً في الخلية؟" إذا كانت الإجابة نعم، فأنت تتجاوز حدود ما يقدمه التمثيل.

Example: Adding a feature of type "gene" with a qualifier gene carrying the name "BRCA1" is a computational representation. Claiming that "this sequence contains the human BRCA1 gene" is a biological claim not proven by the record. The record tells us that the user claimed this location is BRCA1, but it does not prove it. Proving identity requires independent evidence. Practical test: before relying on a feature in a SeqRecord, verify its source (e.g., a reference database) and confirm it with external references.

مثال: إضافة ميزة من نوع "gene" مع مؤهل gene يحمل اسم "BRCA1" هو تمثيل حسابي. الادعاء بأن "هذا التسلسل يحتوي على جين BRCA1 البشري" هو ادعاء بيولوجي لا يثبته السجل. السجل يخبرنا أن المستخدم ادعى أن هذا الموقع هو BRCA1، لكنه لا يثبت ذلك. إثبات الهوية يحتاج إلى أدلة مستقلة. اختبار عملي: قبل أن تعتمد على ميزة في SeqRecord، تحقق من مصدرها (مثل قاعدة بيانات مرجعية) وتأكد من صحتها بمصادر خارجية.

مثال تطبيقي على SeqFeature:

· المدخلات: بداية (start)، نهاية (end)، اتجاه (strand)، نوع (type)، مؤهلات (qualifiers اختياري).

· الافتراضات: أن الإحداثيات صحيحة وتتبع نظام Biopython (zero-based، end exclusive)، وأن الاتجاه معرف (+1، -1، أو 0)، وأن النوع والمؤهلات ذات معنى في السياق البيولوجي.

· المخرجات: كائن SeqFeature يمثل موقعاً على التسلسل.

· القيود: SeqFeature لا يتحقق من أن الموقع يتوافق مع التسلسل، ولا يتحقق من صحة النوع أو المؤهلات، ولا يتحقق من أن الموقع يمثل ميزة حقيقية.

Applied example of SeqFeature:

اختبار: قبل إضافة ميزة، تحقق من أن الإحداثيات ضمن نطاق التسلسل، وأن الاتجاه مناسب للميزة، وأن النوع والمؤهلات متسقة مع المصدر.

· Inputs: start, end, strand, type, qualifiers (optional).

· Assumptions: that the coordinates are correct and follow Biopython's system (zero-based, end exclusive), that the strand is defined (+1, -1, or 0), and that the type and qualifiers are meaningful in the biological context.

· Outputs: a SeqFeature object representing a location on the sequence.

· Limitations: SeqFeature does not verify that the location corresponds to the sequence, nor validate the type or qualifiers, nor verify that the location represents a real feature.

Test: before adding a feature, verify that the coordinates are within the sequence range, that the strand is appropriate for the feature, and that the type and qualifiers are consistent with the source.

عند إضافة ميزات إلى SeqRecord، يجب تسجيل ما يلي:

· المصدر: من أين جاءت معلومات الميزة (قاعدة بيانات، مرجع علمي، تقدير شخصي).

· المعلمات: أي اختيارات في تصميم الميزة (مثل كيفية تحديد الإحداثيات، أو اختيار نوع الميزة).

· التحويلات: أي خطوات قمت بها قبل إضافة الميزة (مثل تحويل إحداثيات من نظام إلى آخر).

When adding features to a SeqRecord, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع فهم من أين جاءت معلومات الميزة،

ويعيد استخدامها في تحليل جديد بثقة؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: where the feature information came from (database, scientific reference, personal estimate).

· Parameters: any choices in feature design (e.g., how to determine coordinates, or choose the feature type).

· Transformations: any steps you performed before adding the feature (e.g., converting coordinates from one system to another).

Test: Practical check: "If I gave this record to a colleague a year later, could they understand where the feature information came from, and reuse it in a new analysis with confidence?" If the answer is no, the documentation is insufficient.

قبل أن تضيف ميزات إلى SeqRecord بناءً على بيانات حقيقية، اختبر آلية إنشاء الميزات على سجل اصطناعي.

أنشئ تسلسلاً صغيراً، وأضف ميزات بإحداثيات معروفة. اكتب السجل إلى ملف GenBank، ثم اقرأه مرة أخرى.

تحقق من أن الميزات بقيت كما هي (البداية، النهاية، الاتجاه، النوع، المؤهلات). إذا نجحت، فأنت واثق من أن آلية

العمل مع SeqFeature سليمة، وأي مشكلة مع بيانات حقيقية تعكس جودة البيانات وليس خطأ في استخدام

Biopython.

Before adding features to a SeqRecord based on real data, test the feature creation mechanism on a synthetic record. Create a small sequence, and add features with known coordinates. Write the record to a GenBank file, then read it back. Verify that the features remain as they were (start, end, strand, type, qualifiers). If this succeeds, the workflow is confident that the project's workflow with SeqFeature is sound, and any problem with real data reflects data quality, not an error in using Biopython.

Chapter 13 — SeqIO: Reading and Writing Files

الفصل 13 — SeqIO: قراءة وكتابة الملفات

الفكرة المركزية: iteration ،parsing، التحقق، تحويل الصيغ، فقد المعلومات.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

SeqIO هو طبقة العبور بين الكائنات البرمجية والملفات. القراءة لا تعني أن الملف أصبح صحيحاً تلقائياً؛ فالبرنامج يستطيع قراءة بنية سليمة تحتوي على بيانات غير مناسبة أو ناقصة. لذلك يجب فحص عدد السجلات، والمعرفات، والأطوال، ونوع الصيغة، وأي بيانات وصفية يعتمد عليها التحليل.

SeqIO is the bridge between program objects and files. Successful parsing does not automatically mean the file is scientifically appropriate; a structurally valid file can still contain incomplete or unsuitable data. Record counts, identifiers, lengths, format, and relevant metadata should therefore be checked.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. ما هو SeqIO؟

1. What Is SeqIO?

SeqIO is a module in Biopython that provides a unified interface for reading and writing multiple sequence file formats. Instead of learning how to read each format separately (FASTA, GenBank, FASTQ, and others), you learn one interface that works with all supported formats. This standardizes the project's workflow and reduces errors from handling different formats.

SeqIO هي وحدة في Biopython توفر واجهة موحدة لقراءة وكتابة صيغ متعددة من ملفات التسلسل. بدلاً من أن تتعلم كيفية قراءة كل صيغة على حدة (FASTA، GenBank، FASTQ، وغيرها)، تتعلم واجهة واحدة تعمل مع جميع الصيغ المدعومة. هذا يوحد طريقة عملك ويقلل من الأخطاء الناتجة عن التعامل مع صيغ مختلفة.

الدالتان الأساسيتان في SeqIO هما:

The two fundamental functions in SeqIO are:

· SeqIO.read(): تقرأ ملفاً يحتوي على سجل واحد (تسلسل واحد) وتعيد كائن SeqRecord.

· SeqIO.read(): Reads a file containing a single record (one sequence) and returns a SeqRecord object.

· SeqIO.parse(): تقرأ ملفاً يحتوي على سجلات متعددة (عدة تسلسلات) وتعيد مكرراً (iterator) من كائنات SeqRecord.

· SeqIO.parse(): Reads a file containing multiple records (multiple sequences) and returns an iterator of SeqRecord objects.

```
from Bio import SeqIO
# قراءة ملف FASTA يحتوي على تسلسل واحد
record = SeqIO.read("single_sequence.fasta", "fasta")
print(f"المعرف: {record.id}")
# قراءة ملف FASTA يحتوي على عدة تسلسلات
for record in SeqIO.parse("multiple_sequences.fasta", "fasta"):
    print(f"المعرف: {record.id}, الطول: {len(record.seq)}")
```

2. Single Read (read) vs. Iterative Processing (parse)

SeqIO.read() is appropriate when you know the file contains only one record. If you use it on a file containing multiple records, you will get an error message. SeqIO.parse() is suitable for large files containing multiple records, because it returns an iterator rather than loading all records into memory at once. This allows efficient processing of large files without excessive memory consumption.

2. القراءة المفردة (read) مقابل المعالجة المتتابعة (parse)

SeqIO.read() مناسبة عندما تعرف أن الملف يحتوي على سجل واحد فقط. إذا استخدمتها على ملف يحتوي على عدة سجلات، ستظهر لك رسالة خطأ. SeqIO.parse() مناسبة للملفات الكبيرة التي تحتوي على عدة سجلات، لأنها تعيد مكرراً (iterator) بدلاً من تحميل كل السجلات في الذاكرة دفعة واحدة. هذا يسمح بمعالجة الملفات الكبيرة بكفاءة، دون استهلاك ذاكرة زائدة.

```
# ملف كبير يحتوي على آلاف التسلسلات
# المعالجة المتتابعة (تقرأ سجلاً واحداً في كل مرة)
for record in SeqIO.parse("large_file.fasta", "fasta"):
    # معالجة كل سجل على حدة
    if len(record.seq) > 1000:
        print(f"تسلسل طويل: {record.id}")
```

Choosing read or parse is not just a technical detail; it is a decision that affects the project's program's performance and scalability. For small files, both methods work. For large files, parse is essential.

اختيار read أو parse ليس مجرد تفصيل تقني؛ إنه قرار يؤثر على أداء برنامجك وقابليته للتوسع. للملفات الصغيرة، كلا الطريقتين تعملان. للملفات الكبيرة، parse ضرورية.

3. Choosing the Appropriate Format

Not every format is suitable for every purpose. Format selection depends on what the analysis requires from the data:

3. اختيار الصيغة المناسبة (Format Selection)

ليس كل صيغة مناسبة لكل غرض. اختيار الصيغة يعتمد على ما تحتاجه من البيانات:

· FASTA: أبسط صيغة، تحوي فقط المعرف والوصف والتسلسل. مناسبة للبحث السريع، والمقارنات البسيطة، والمشاركة السهلة. لكنها لا تحمل بيانات وصفية معقدة أو ميزات.

· FASTA: The simplest format, containing only the identifier, description, and sequence. Suitable for quick searching, simple comparisons, and easy sharing. However, it does not carry complex metadata or features.

· GenBank: صيغة غنية تحوي المعرف، الوصف، التسلسل، البيانات الوصفية (annotations)، والميزات (features). مناسبة للتوثيق الكامل والتحليل المتقدم. لكن الملفات أكبر حجماً.

· GenBank: A rich format containing the identifier, description, sequence, annotations, and features. Suitable for complete documentation and advanced analysis. However, files are larger.

· FASTQ: صيغة تحوي التسلسل بالإضافة إلى درجات جودة لكل قاعدة (quality scores). مناسبة لبيانات التسلسل عالية الإنتاجية (NGS) قبل المعالجة.

· FASTQ: A format containing the sequence along with quality scores for each base. Suitable for high-throughput sequencing (NGS) data before processing.

```
#قراءة ملف GenBank للحصول على معلومات غنية
record = SeqIO.read("gene.gb", "genbank")
print(f"المعرف: {record.id}")
print(f"البيانات الوصفية: {record.annotations}")
print(f"عدد الميزات: {len(record.features)}")
#قراءة ملف FASTQ للحصول على جودة القراءات
for record in SeqIO.parse("sample.fastq", "fastq"):
  print(f"جودة القواعد: {record.letter_annotations['phred_quality']}")
Rule: Choose the format that carries the minimum sufficient information for the project's analysis, but do not choose a format that loses information you will later need.
```

4. Format Conversion and Information Loss

القاعدة: اختر الصيغة التي تحمل أقل معلومات كافية لتحليلك، ولكن لا تختار صيغة تفقد معلومات تحتاجها لاحقاً.

When converting a file from one format to another, you may lose important information. Converting GenBank to FASTA loses features and metadata. Converting FASTQ to FASTA loses quality scores. These losses are not a flaw in Biopython; they are a consequence of the limitations of the formats themselves. FASTA simply cannot store features, and FASTQ cannot store complex metadata. Therefore, before any conversion, Practical check: Will I need this information later? If the answer is yes, keep the original file or use a format that preserves the information.

4. تحويل الصيغ وفقد المعلومات

عند تحويل ملف من صيغة إلى أخرى، قد تفقد معلومات مهمة. تحويل GenBank إلى FASTA يفقد الميزات والبيانات الوصفية. تحويل FASTQ إلى FASTA يفقد درجات الجودة. هذه الفقدانات ليست عيباً في Biopython؛ إنها نتيجة لقيود الصيغ نفسها. FASTA ببساطة لا يمكن تخزين الميزات، و FASTQ لا يمكن تخزين البيانات الوصفية المعقدة. لذلك، قبل أي تحويل، فحص تطبيقي: هل سأحتاج هذه المعلومات لاحقاً؟ إذا كانت الإجابة نعم، فاحتفظ بالملف الأصلي أو استخدم صيغة تحافظ على المعلومات.

```
# تحويل GenBank إلى FASTA يفقد الميزات
gb_record = SeqIO.read("gene.gb", "genbank")
SeqIO.write(gb_record, "gene.fasta", "fasta")  # الميزات والبيانات الوصفية لا تظهر في ملف FASTA
```

```
# تحويل FASTQ إلى FASTA يفقد درجات الجودة
fastq_record = SeqIO.read("sample.fastq", "fastq")
SeqIO.write(fastq_record, "sample.fasta", "fasta") # درجات الجودة لا تظهر في ملف FASTA
```

This does not mean that conversion is forbidden; it means it should be a conscious decision. If the analysis requires feature information, use GenBank. If the analysis requires quality, use FASTQ. If you only need the sequence, use FASTA.

هذا لا يعني أن التحويل ممنوع، بل يعني أنه يجب أن يكون قراراً واعياً. إذا كنت بحاجة إلى معلومات الميزات، استخدم GenBank. إذا كنت بحاجة إلى الجودة، استخدم FASTQ. إذا كنت بحاجة فقط إلى التسلسل، استخدم FASTA.

5. Checking Inputs and Outputs

Before starting the project's analysis, check the file the workflow is reading. Check: How many records are in the file? Is the format correct? Are the fields as expected? After reading, check the results: Is the number of records as expected? Is the metadata present? Checking inputs and outputs protects you from silent errors that could corrupt the project's analysis.

5. التحقق من المدخلات والمخرجات

قبل أن تبدأ تحليلك، تحقق من الملف الذي تقرؤه. فحص: كم عدد السجلات في الملف؟ هل الصيغة صحيحة؟ هل الحقول متوقعة؟ بعد القراءة، تحقق من النتائج: هل عدد السجلات كما هو متوقع؟ هل البيانات الوصفية موجودة؟

التحقق من المدخلات والمخرجات يحميك من الأخطاء الصامتة التي قد تفسد تحليلك.

```
# التحقق من عدد السجلات قبل المعالجة
records = list(SeqIO.parse("data.fasta", "fasta"))
print(f"عدد السجلات: {len(records)}")
# التحقق من وجود معرفات فريدة
ids = [record.id for record in records]
if len(ids) != len(set(ids)):
    print("تحذير: توجد معرفات مكررة!")
# التحقق من وجود البيانات الوصفية المتوقعة
for record in records:
    if "organism" not in record.annotations:
```

print(f"تحذير: السجل {record.id} لا يحتوي على بيانات عن الكائن الحي")

6. Practical Application: Reading and Writing Multiple Files

In a practical application, you might want to read multiple files, process each, and write the results to a new file. This is a common pattern in pipelines:

6. التطبيق العملي: قراءة وكتابة ملفات متعددة

في تطبيق عملي، قد تريد قراءة ملفات متعددة، معالجة كل منها، وكتابة النتائج إلى ملف جديد. هذا مثال شائع في خطوط العمل (pipelines):

```
from Bio import SeqIO
import os
def process_fasta_files(input_directory, output_file):
    """قراءة جميع ملفات FASTA في مجلد، وحساب نسبة GC، وكتابة النتائج."""
    results = []
    for filename in os.listdir(input_directory):
        if not filename.endswith(".fasta"):
            continue
        filepath = os.path.join(input_directory, filename)
        for record in SeqIO.parse(filepath, "fasta"):
            gc = gc_fraction(record.seq)*100 # حساب نسبة GC (تحتاج إلى استيراد GC من Bio.SeqUtils)
            results.append({
                "file": filename,
                "id": record.id,
```

```
                "length": len(record.seq),
                "gc": gc
            })
#كتابة النتائج إلى ملف
    with open(output_file, "w") as f:
        f.write("file,id,length,gc\n")
        for row in results:
            f.write(f"{row['file']},{row['id']},{row['length']},{row['gc']:.2f}\n")
return results
```

في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

7. حدود SeqIO

7. Limitations of SeqIO

SeqIO supports many formats, but it does not support every existing format. You may encounter rare formats or formats specific to a particular institution. In this case, you may need to write the project's own parser or use other tools. Also, SeqIO does not directly handle compressed files (e.g., .gz); the analysis requires to decompress before reading. These limitations are not flaws; they are a reminder that Biopython is a powerful tool but not comprehensive for everything.

SeqIO تدعم العديد من الصيغ، لكنها لا تدعم كل الصيغ الموجودة. قد تواجه صيغاً نادرة أو خاصة بمؤسسة معينة. في هذه الحالة، قد تحتاج إلى كتابة محلل (parser) خاص بك، أو استخدام أدوات أخرى. أيضاً، SeqIO لا تتعامل مع الملفات المضغوطة مباشرة (مثل .gz)؛ تحتاج إلى فك الضغط قبل القراءة. هذه القيود ليست عيباً؛ إنها تذكير بأن Biopython أداة قوية لكنها ليست شاملة لكل شيء.

Chapter Summary

خلاصة الفصل

SeqIO is the unified interface for reading and writing sequence files in Biopython. It provides read() for single records and parse() for multiple records. Choosing the appropriate format depends on what the analysis requires from the data: FASTA for simplicity, GenBank for richness, FASTQ for quality. Converting between formats may lead to information loss, so it should be a conscious decision. Checking inputs and outputs is an essential part of the process, to protect the analysis from silent errors. The next chapter will delve into the three most common formats: FASTA, FASTQ, and GenBank.

SeqIO هي الواجهة الموحدة لقراءة وكتابة ملفات التسلسل في Biopython. توفر read() للسجلات المفردة و parse() للسجلات المتعددة. اختيار الصيغة المناسبة يعتمد على ما تحتاجه من البيانات: FASTA للبساطة، GenBank للغنى، FASTQ للجودة. التحويل بين الصيغ قد يؤدي إلى فقدان المعلومات، لذلك يجب أن يكون قراراً واعياً. التحقق من المدخلات والمخرجات هو جزء أساسي من العملية، لحماية التحليل من الأخطاء الصامتة. الفصل القادم سيتعمق في الصيغ الثلاث الأكثر شيوعاً: FASTQ ،FASTA، و GenBank.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "SeqIO: Reading and Writing Files" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "SeqIO: قراءة وكتابة الملفات" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "SeqIO can read and write any existing file format." This is incorrect; SeqIO supports many formats but not all. The correct statement is: "SeqIO provides a unified interface for many common sequence file formats, but it does not support every format, and you may need additional tools for rare formats." A simple test: Practical check: "Is the format I need supported in SeqIO?" Check the official documentation to confirm.

مثال على التجاوز: القول إن "SeqIO يمكن قراءة وكتابة أي صيغة ملفات موجودة". هذا غير صحيح؛ SeqIO تدعم العديد من الصيغ لكن ليس كلها. الصحيح هو: "SeqIO توفر واجهة موحدة للعديد من صيغ ملفات التسلسل الشائعة، لكنها لا تدعم كل الصيغ، وقد تحتاج إلى أدوات إضافية للصيغ النادرة." اختبار بسيط: فحص تطبيقي: "هل الصيغة التي أحتاجها مدعومة في SeqIO؟" تحقق من التوثيق الرسمي للتأكد.

Example: Reading a FASTA file and converting it to a SeqRecord is a computational representation. Claiming that "this record represents a real gene from the mentioned organism" is a biological claim not proven by merely reading the file. The file tells us what its creator wrote, but it does not prove its biological correctness. Practical test: before relying on a record from a file, verify its source and confirm it with external references.

مثال: قراءة ملف FASTA وتحويله إلى SeqRecord هو تمثيل حسابي. الادعاء بأن "هذا السجل يمثل جيناً حقيقياً من الكائن الحي المذكور" هو ادعاء بيولوجي لا يثبته مجرد قراءة الملف. الملف يخبرنا بما كتبه منشئه، لكنه لا يثبت صحته البيولوجية. اختبار عملي: قبل أن تعتمد على سجل من ملف، تحقق من مصدره وتأكد من صحته بمصادر خارجية.

مثال تطبيقي على SeqIO.read():

· المدخلات: مسار ملف، والصيغة (مثل "fasta").

· الافتراضات: أن الملف موجود وقابل للقراءة، وأنه يحتوي على سجل واحد صحيح، وأن الصيغة المحددة تتطابق مع محتوى الملف.

· المخرجات: كائن SeqRecord واحد.

· القيود: إذا كان الملف يحتوي على عدة سجلات، ستظهر رسالة خطأ. إذا كانت الصيغة غير صحيحة، قد يظهر خطأ أو قد يُقرأ الملف بشكل خاطئ.

Applied example of SeqIO.read():

اختبار: قبل استخدام read، تأكد من أن الملف يحتوي على سجل واحد فقط، أو استخدم parse بدلاً من ذلك.

· Inputs: a file path, and the format (e.g., "fasta").

· Assumptions: that the file exists and is readable, that it contains one valid record, and that the specified format matches the file content.

· Outputs: a single SeqRecord object.

· Limitations: if the file contains multiple records, an error message will appear. If the format is incorrect, an error may appear or the file may be read incorrectly.

Test: before using read, ensure the file contains only one record, or use parse instead.

عند استخدام SeqIO، يجب تسجيل ما يلي:

· المصدر: الملف الأصلي (المسار، الاسم، التاريخ).

· الإصدار: إصدار Biopython المستخدم.

· المعلمات: الصيغة المستخدمة (مثل "genbank"، "fasta")، وأي معاملات إضافية.

· التحويلات: أي خطوات قمت بها بعد القراءة (مثل تحويل الصيغة، أو استخراج جزء من البيانات).

When using SeqIO, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا الكود والملف لزميل بعد عام، هل يستطيع إعادة نفس القراءة والمعالجة دون تغييرات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the original file (path, name, date).

· Version: the Biopython version used.

· Parameters: the format used (e.g., "fasta", "genbank"), and any additional parameters.

· Transformations: any steps you performed after reading (e.g., converting format, or extracting part of the data).

Test: Practical check: "If I gave this code and file to a colleague a year later, could they reproduce the same reading and processing without changes?" If the answer is no, the documentation is insufficient.

قبل أن تستخدم SeqIO على بيانات حقيقية، اختبر قراءة وكتابة الملفات على ملفات اصطناعية صغيرة. أنشئ ملف FASTA بسيطاً يحتوي على تسلسل واحد، واقرأه باستخدام SeqIO.read(). ثم أنشئ ملفاً يحتوي على عدة تسلسلات، واقرأه باستخدام SeqIO.parse(). اكتب السجلات التي قرأتها إلى ملف جديد، واقرأه مرة أخرى. تحقق من أن الهوية والوصف والتسلسل بقيت كما هي. إذا نجحت هذه الاختبارات، فأنت واثق من أن عملك مع SeqIO سليم، وأي مشكلة مع بيانات حقيقية تعكس جودة البيانات وليس خطأ في استخدامك للمكتبة.

Before using SeqIO on real data, test reading and writing on small synthetic files. Create a simple FASTA file containing a single sequence, and read it using SeqIO.read(). Then create a file containing multiple sequences, and read it using SeqIO.parse(). Write the records you read to a new file, and read it back. Verify that the identity, description, and sequence remain as they were. If these tests succeed, the workflow is confident that the project's workflow with SeqIO is sound, and any problem with real data reflects data quality, not an error in using the library.

Chapter 14 — FASTA, FASTQ, and GenBank

الفصل 14 — FASTA و FASTQ و GenBank

الفكرة المركزية: طبقات المعلومات، الجودة، annotations، دلالة الصيغة.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

الاختلاف بين الصيغ ليس اختلاف امتداد الملف فقط. FASTA يحفظ التسلسل مع معرف، وFASTQ يضيف جودة لكل قاعدة، بينما GenBank يحمل طبقة أوسع من البيانات الوصفية والميزات. تحويل صيغة إلى أخرى قد يكون فقداً للمعلومات، لذلك يجب أن يُسجل التحويل ويُعرف ما الذي بقي وما الذي اختفى.

The difference among these formats is not merely a file-extension difference. FASTA stores sequence and identifier, FASTQ adds per-base quality, and GenBank carries a richer annotation layer. Converting between formats can therefore be an information-losing operation, and the transformation should document what was preserved and what was lost.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. FASTA: Simplicity and Widespread Use

1. FASTA: البساطة والانتشار

FASTA هي أبسط صيغة وأكثرها انتشاراً لتمثيل التسلسلات. تتكون من سطرين أساسيين:

FASTA is the simplest and most widely used format for representing sequences. It consists of two basic lines:

· سطر الرأس (Header): يبدأ بعلامة > ويحمل المعرف (ID) ووصفاً مختصراً.

· The header line: starts with > and carries the identifier (ID) and a brief description.

· سطر التسلسل: يحمل التسلسل نفسه (DNA، RNA، أو بروتين).

· The sequence line: carries the sequence itself (DNA, RNA, or protein).

Language / نوع الكود: **FASTA/FASTQ**

Purpose: show the structure of a synthetic sequence format and distinguish sequence from accompanying metadata or quality.

الغرض: عرض بنية صيغة تسلسل اصطناعية وتوضيح الفرق بين التسلسل والبيانات المرافقة له.

```
>SYN_001 Synthetic gene for exercise
ATGCGTACGTAGCTAGCTAGCTAGCTAGC
```

ما تحمله FASTA:

· وصف مختصر (Description)

· المعرف (ID)

· التسلسل

ما لا تحمله:

· درجات الجودة (Quality scores).

· البيانات الوصفية المنظمة (Structured metadata).

What FASTA carries:

· الميزات الموضعية (Features مثل الجينات والإكسونات).

· Identifier (ID)

· الإحداثيات أو الاتجاه

· Brief description

· The sequence

What FASTA does not carry:

· Quality scores

· Structured metadata

· Localized features (such as genes and exons).

· Coordinates or strand information.

FASTA is suitable for quick searching, simple comparisons, and easy sharing of sequences. However, it is not suitable for analyses that require additional information about the sequence.

FASTA مناسبة للبحث السريع، والمقارنات البسيطة، ومشاركة التسلسلات بسهولة. لكنها غير مناسبة للتحليلات

التي تحتاج إلى معلومات إضافية عن التسلسل.

2. FASTQ: Quality per Base

2. FASTQ: الجودة لكل قاعدة

FASTQ هي الصيغة القياسية لبيانات التسلسل عالية الإنتاجية (NGS). تحمل التسلسل بالإضافة إلى درجة جودة

لكل قاعدة. تتكون من أربعة أسطر لكل قراءة (read):

· السطر الأول: يبدأ بعلامة @ ويحمل المعرف.

· السطر الثاني: التسلسل.

FASTQ is the standard format for high-throughput sequencing (NGS) data. It carries the sequence along with a quality score for each base. It consists of four lines per read:

· السطر الثالث: يبدأ بعلامة + وقد يحمل المعرف (غالباً ما يكون فارغاً).

· The first line: starts with @ and carries the identifier.

· السطر الرابع: سلسلة من الرموز تمثل درجات الجودة (Quality scores) لكل قاعدة في التسلسل.

· The second line: the sequence.

· The third line: starts with + and may carry the identifier (often empty).

· The fourth line: a string of symbols representing quality scores for each base in the sequence.

Language / نوع الكود: **FASTA/FASTQ**

```
@SEQ_ID_001
ATGCGTACGTAGCTAGCTAGCTAGCTAGC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

ما تحمله FASTQ:

· المعرف (ID)

· التسلسل

· درجة جودة لكل قاعدة (Quality score per base).

· معلومات تقنية إضافية (في السطر الأول غالباً).

ما لا تحمله FASTQ:

· بيانات وصفية منظمة عن العينة أو الكائن الحي.

What FASTQ carries:

· ميزات موضعية

· Identifier (ID)

· إحداثيات

· The sequence

· A quality score for each base.

· Additional technical information (often in the first line).

What FASTQ does not carry:

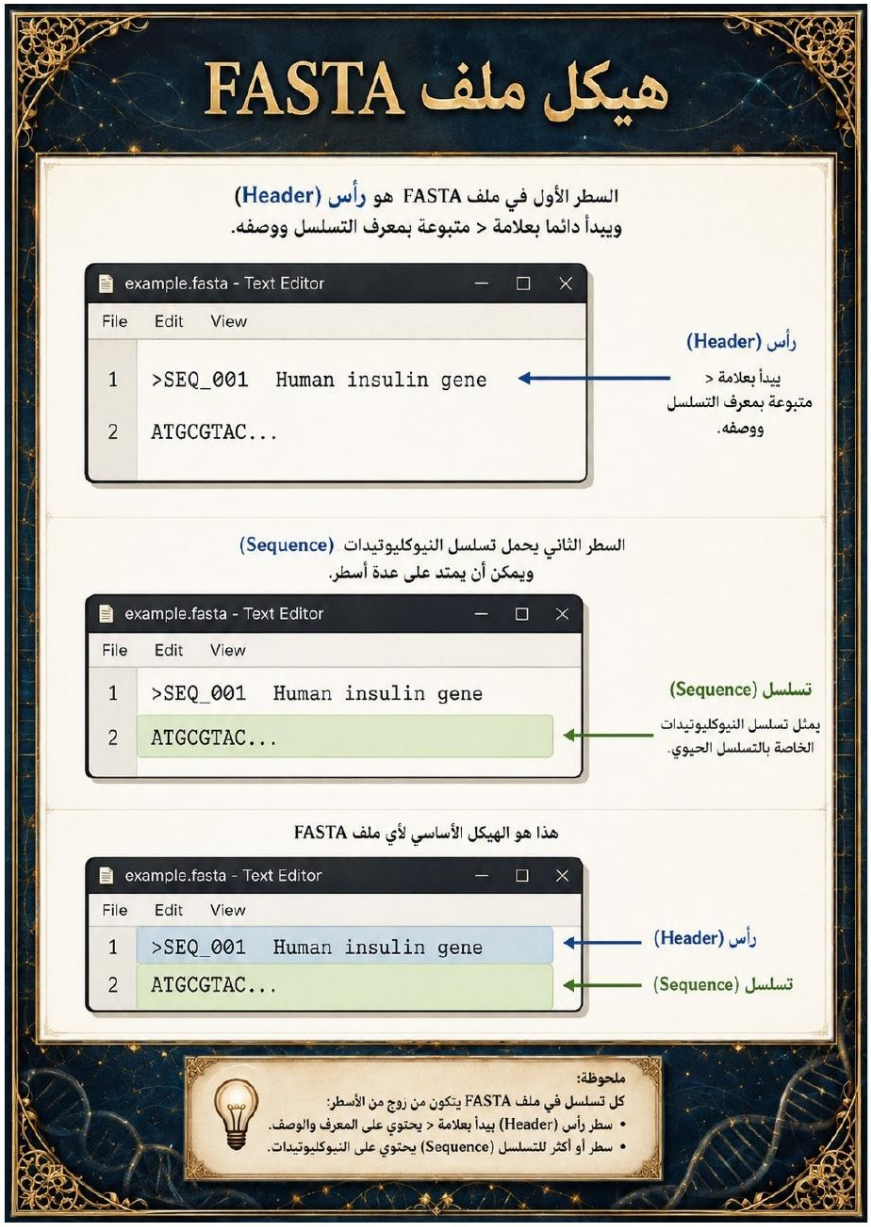
· Structured metadata about the sample or organism.

· Localized features

· Coordinates

Quality scores in FASTQ express the sequencer's confidence in the correctness of each base. A Phred score of 20 means a 1% error probability (1 in 100), and a Phred score of 30 means a 0.1% error probability (1 in 1000). These scores are critical for filtering low-quality reads before analysis.

درجات الجودة في FASTQ تُعبر عن ثقة جهاز التسلسل في صحة كل قاعدة. درجة Phred 20 تعني احتمال خطأ 1% (1 من 100)، ودرجة Phred 30 تعني احتمال خطأ 0.1% (1 من 1000). هذه الدرجات حاسمة لتصفية القراءات منخفضة الجودة قبل التحليل.


هيكل ملف FASTA
السطر الأول في ملف FASTA هو رأس (Header)
ويبدأ دائما بعلامة > متبوعة بمعرف التسلسل ووصفه.
example.fasta - Text Editor
File Edit View
1 >SEQ_001 Human insulin gene
2 ATGCGTAC...
رأس (Header)
يبدأ بعلامة > متبوعة بمعرف التسلسل ووصفه.
السطر الثاني يحمل تسلسل النيوكليوتيدات (Sequence)
ويمكن أن يمتد على عدة أسطر.
example.fasta - Text Editor
File Edit View
1 >SEQ_001 Human insulin gene
2 ATGCGTAC...
تسلسل (Sequence)
يمثل تسلسل النيوكليوتيدات الخاصة بالتسلسل الحيوي.
هذا هو الهيكل الأساسي لأي ملف FASTA
example.fasta - Text Editor
File Edit View
1 >SEQ_001 Human insulin gene
2 ATGCGTAC...
رأس (Header)
تسلسل (Sequence)
ملحوظة:
كل تسلسل في ملف FASTA يتكون من زوج من الأسطر:
• سطر رأس (Header) يبدأ بعلامة > يحتوي على المعرف والوصف.
• سطر أو أكثر للتسلسل (Sequence) يحتوي على النيوكليوتيدات.

Infographic 1 / إنفوجرافيك 1

3. GenBank: Richness and Complete Documentation

3. GenBank: الغنى والتوثيق الكامل

GenBank is a rich format designed to fully document sequences. It carries the sequence, identifier, description, structured metadata (annotations), localized features, and scientific references. It is the standard format for databases such as GenBank, EMBL, and DDBJ.

GenBank هي صيغة غنية صُممت لتوثيق التسلسلات بشكل كامل. تحمل التسلسل، المعرف، الوصف، البيانات الوصفية المنظمة (annotations)، الميزات الموضعية (features)، والمراجع العلمية. هي الصيغة المعيارية لقواعد البيانات مثل GenBank و EMBL و DDBJ.

GenBank :نوع الكود / Language

Purpose: show how GenBank carries sequence together with richer identity, description, and feature information.

الغرض: إظهار كيف تحمل صيغة GenBank التسلسل مع طبقات إضافية من الهوية والوصف والميزات.

```
LOCUS       SYN_001                30 bp    DNA     linear   SYN 15-JAN-2025
DEFINITION  Synthetic gene for exercise.
ACCESSION   SYN_001
VERSION     SYN_001.1
KEYWORDS    synthetic.
SOURCE      synthetic construct
ORGANISM  synthetic construct
...
FEATURES             Location/Qualifiers
     gene            1..30
                     /gene="SYN_GENE_001"
     CDS             1..30
                     /gene="SYN_GENE_001"
                     /codon_start=1
                     /product="hypothetical protein"
ORIGIN
```

```
1 ATGCGTACGTAGCTAGCTAGCTAGCTAGC
//
```

ما تحمله GenBank:

· المعرف (Accession number)

· الوصف (Definition)

· التسلسل

· البيانات الوصفية المنظمة (الكائن الحي، المصدر، التاريخ).

· الميزات الموضعية (Features) مع الإحداثيات والاتجاه.

· المراجع العلمية

· معلومات الإصدار

What GenBank carries:

ما لا تحمله GenBank (بشكل مباشر):

· درجات الجودة لكل قاعدة (هذه موجودة في FASTQ).

· Accession number

· Definition

· The sequence

· Structured metadata (organism, source, date).

· Localized features with coordinates and strand.

· Scientific references

· Version information

What GenBank does not directly carry:

· Per-base quality scores (these are in FASTQ).

GenBank هي الصيغة المثالية للتوثيق الدائم والتحليل المتقدم، حيث تحمل كل المعلومات اللازمة لتفسير التسلسل في سياقه البيولوجي.

GenBank is the ideal format for permanent documentation and advanced analysis, carrying all the information needed to interpret the sequence in its biological context.

4. Comparison Between Formats

الميزة FASTA FASTQ GenBank

4. مقارنة بين الصيغ

المعرف والوصف نعم نعم نعم

التسلسل نعم نعم نعم

درجات الجودة لا نعم لا

البيانات الوصفية لا محدودة نعم (منظمة).

Feature FASTA FASTQ GenBank

الميزات الموضعية لا لا نعم

Identifier and description Yes Yes Yes.

المراجع العلمية لا لا نعم

حجم الملف صغير متوسط كبير

Sequence Yes Yes Yes

Quality scores No Yes No

Metadata No Limited Yes (structured).

Localized features No No Yes

Scientific references No No Yes.

File size Small Medium Large

لا توجد صيغة "أفضل" دائماً. الاختيار يعتمد على السؤال العلمي وسير العمل:

· FASTA: للمقارنات السريعة ومشاركة التسلسلات.

There is no "best" format always. The choice depends on the scientific question and workflow:

· FASTQ: لبيانات التسلسل الخام قبل المعالجة.

· FASTA: For quick comparisons and sharing sequences.

· GenBank: للتوثيق الكامل والتحليل المتقدم.

· FASTQ: For raw sequencing data before processing.

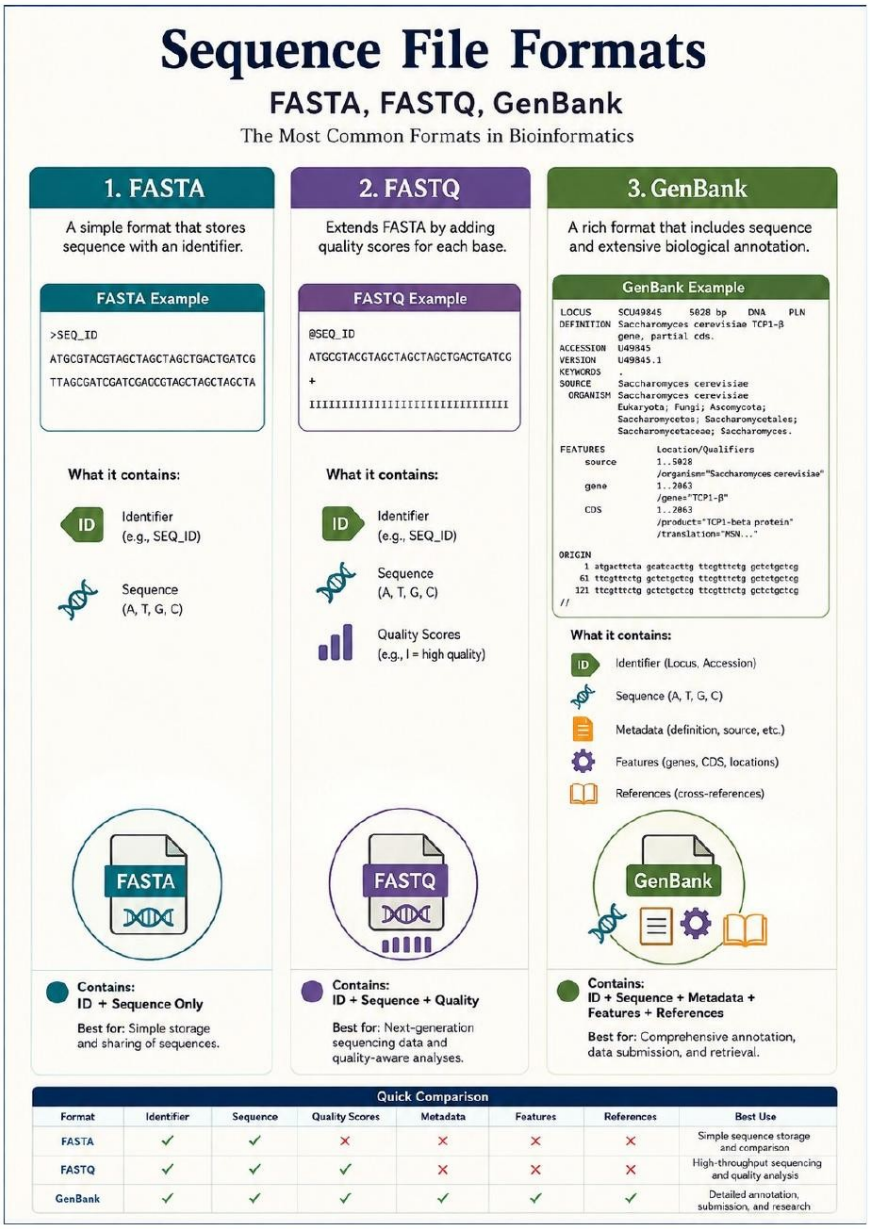
· GenBank: For complete documentation and advanced analysis.

5. Format Semantics

5. دلالة الصيغة (Format Semantics)

Format semantics means "what does each field in this format mean?" Understanding format semantics is what prevents you from misinterpreting the data. In FASTA, the header is just free text; there is no precise definition of what it should contain. In FASTQ, the quality score has a precise mathematical meaning (error probability). In GenBank, each field has a specific definition in the official specification. When converting between formats, the workflow is not just changing the file extension; the workflow is changing what can be represented and what cannot.

دلالة الصيغة تعني "ماذا يعني كل حقل في هذه الصيغة؟" فهم دلالة الصيغة هو ما يمنعك من سوء تفسير البيانات. في FASTA، سطر الرأس هو مجرد نص حر؛ لا يوجد تعريف دقيق لما يجب أن يحتويه. في FASTQ، درجة الجودة لها معنى رياضي دقيق (احتمال الخطأ). في GenBank، كل حقل له تعريف محدد في المواصفات الرسمية. عند التحويل بين الصيغ، أنت لا تغير فقط امتداد الملف؛ أنت تغير ما يمكن تمثيله وما لا يمكن تمثيله.


Sequence File Formats
FASTA, FASTQ, GenBank
The Most Common Formats in Bioinformatics
1. FASTA
A simple format that stores sequence with an identifier.
FASTA Example
>SEQ_ID
ATGCGTACGTAGCTAGCTAGCTGACTGATCG
TTAGCGATCGATCGACCGTAGCTAGCTAGCTA
What it contains:
ID
Identifier (e.g., SEQ_ID)
Sequence (A, T, G, C)
FASTA
Contains: ID + Sequence Only
Best for: Simple storage and sharing of sequences.
2. FASTQ
Extends FASTA by adding quality scores for each base.
FASTQ Example
@SEQ_ID
ATGCGTACGTAGCTAGCTAGCTGACTGATCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
What it contains:
ID
Identifier (e.g., SEQ_ID)
Sequence (A, T, G, C)
Quality Scores (e.g., I = high quality)
FASTQ
Contains: ID + Sequence + Quality
Best for: Next-generation sequencing data and quality-aware analyses.
3. GenBank
A rich format that includes sequence and extensive biological annotation.
GenBank Example
LOCUS SCU49845 5028 bp DNA PLN
DEFINITION Saccharomyces cerevisiae TCP1-β gene, partial cds.
ACCESSION U49845
VERSION U49845.1
KEYWORDS .
SOURCE Saccharomyces cerevisiae
ORGANISM Saccharomyces cerevisiae
Eukaryota; Fungi; Ascomycota; Saccharomycetes; Saccharomycetales; Saccharomycetaceae; Saccharomyces.
FEATURES Location/Qualifiers
source 1..5028
/organism="Saccharomyces cerevisiae"
gene 1..2863
/gene="TCP1-β"
CDS 1..2863
/product="TCP1-beta protein"
/translation="MSN..."
ORIGIN
1 atgacttcta gcatcacttg ttcgtttctg gctctgctcg
61 ttcgttcctg gctctgctcg ttcgtttctg gctctgctcg
121 ttcgttcctg gctctgctcg ttcgtttctg gctctgctcg
//
What it contains:
ID
Identifier (Locus, Accession)
Sequence (A, T, G, C)
Metadata (definition, source, etc.)
Features (genes, CDS, locations)
References (cross-references)
GenBank
Contains: ID + Sequence + Metadata + Features + References
Best for: Comprehensive annotation, data submission, and retrieval.
Quick Comparison
Format | Identifier | Sequence | Quality Scores | Metadata | Features | References | Best Use
FASTA | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | Simple sequence storage and comparison
FASTQ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | High-throughput sequencing and quality analysis
GenBank | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | Detailed annotation, submission, and research

Infographic 25 / إنفوجرافيك 25


```
#تحويل GenBank إلى FASTA يفقد الميزات والبيانات الوصفية
#هذا هو ما تعنيه الصيغتان Biopython هذا ليس خطأ في ؛
gb_record = SeqIO.read("gene.gb", "genbank")
SeqIO.write(gb_record, "gene.fasta", "fasta") #الميزات تختفي
```

6. Information Loss During Conversion

When converting from a rich format to a simpler one, you lose information. This loss is not always a problem, but it is appropriate to be aware of it:

6. فقد المعلومات أثناء التحويل

عند التحويل من صيغة غنية إلى صيغة أبسط، تفقد معلومات. هذه الخسارة ليست دائماً مشكلة، لكن يجب أن تكون واعياً لها:

· GenBank → FASTA: تفقد البيانات الوصفية (annotations) والميزات (features) والمراجع. يبقى المعرف والوصف والتسلسل فقط.

· GenBank → FASTA: You lose annotations, features, and references. Only the identifier, description, and sequence remain.

· FASTQ → FASTA: تفقد درجات الجودة لكل قاعدة. يبقى المعرف والتسلسل فقط.

· FASTQ → FASTA: You lose per-base quality scores. Only the identifier and sequence remain.

· GenBank → FASTQ: غير ممكن مباشرة لأن GenBank لا يحتوي على درجات جودة.

· GenBank → FASTQ: Not directly possible because GenBank does not contain quality scores.

The golden rule: keep the original file. If the analysis requires to convert for a specific tool, keep a copy of the original file to refer back to when the analysis requires the lost information.

القاعدة الذهبية: احتفظ بالملف الأصلي. إذا كنت بحاجة إلى التحويل لاستخدام أداة معينة، فاحتفظ بنسخة من الملف الأصلي للرجوع إليها عندما تحتاج المعلومات المفقودة.

```
# تحويل FASTQ إلى FASTA مع فقدان الجودة
fastq_record = SeqIO.read("sample.fastq", "fastq")
SeqIO.write(fastq_record, "sample.fasta", "fasta") # تفقد درجات الجودة
# إذا كنت بحاجة إلى الجودة لاحقاً، احتفظ بالملف الأصلي
```

7. Practical Application: Choosing the Appropriate Format

For illustration, consider a case in which have three different scenarios:

7. التطبيق العملي: اختيار الصيغة المناسبة

Scenario 1: You want to compare a sequence with other sequences in a database. You only need the sequence and identifier. → Use FASTA.

تخيل أن يوجد في المثال ثلاثة سيناريوهات مختلفة:

Scenario 2: the workflow has raw sequencing data from an Illumina machine, and you want to filter low-quality reads before alignment. the analysis requires quality scores. → Use FASTQ.

السيناريو الأول: تريد مقارنة تسلسل مع تسلسلات أخرى في قاعدة بيانات. تحتاج فقط إلى التسلسل والمعرف. → استخدم FASTA.

Scenario 3: You want to publish a complete gene sequence with all its information (gene, exons, protein product, references). → Use GenBank.

السيناريو الثاني: يوجد في المثال بيانات تسلسل خام من جهاز Illumina، وتريد تصفية القراءات منخفضة الجودة قبل المحاذاة. تحتاج إلى درجات الجودة. → استخدم FASTQ.

السيناريو الثالث: تريد نشر تسلسل جين كامل مع جميع المعلومات عنه (الجين، الإكسونات، البروتين المنتج، المراجع). → استخدم GenBank.

Chapter Summary

خلاصة الفصل

FASTA, FASTQ, and GenBank are the three most common formats in bioinformatics. FASTA is simple and carries only the sequence and identifier. FASTQ adds per-base quality scores, essential for raw sequencing data. GenBank is the richest format, carrying metadata, features, and references. There is no best format always; the choice depends on the scientific question and workflow. Converting between formats may lead to loss of important information, so it should be a conscious decision, and the original file should be kept for reference. The next chapter will move from data representation to data analysis, with an introduction to alignment and similarity.

FASTA، FASTQ، و GenBank هي الصيغ الثلاث الأكثر شيوعاً في المعلوماتية الحيوية. FASTA بسيطة وتحمل التسلسل والمعرف فقط. FASTQ تضيف درجات الجودة لكل قاعدة، وهي ضرورية لبيانات التسلسل الخام. GenBank هي الصيغة الأغنى، وتحمل البيانات الوصفية والميزات والمراجع. لا توجد صيغة أفضل دائماً؛ الاختيار يعتمد على السؤال العلمي وسير العمل. التحويل بين الصيغ قد يؤدي إلى فقدان معلومات مهمة، لذلك يجب أن يكون قراراً واعياً، ويجب الاحتفاظ بالملف الأصلي للرجوع إليه. الفصل القادم سينتقل بك من تمثيل البيانات إلى تحليلها، مع مقدمة عن المحاذاة والتشابه.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "FASTA, FASTQ, and GenBank" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "FASTA و FASTQ و GenBank" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "GenBank contains all possible information about the sequence." This is incorrect; GenBank contains a lot of information, but it does not contain everything (e.g., three-dimensional structure, quality scores, molecular interactions). The correct statement is: "GenBank is the richest among the common formats, carrying metadata, features, and references, but it does not represent the complete model of the living molecule." A simple test: Practical check: "Does this format carry everything I need for my current analysis?" and not "Does it carry everything possible?".

مثال على التجاوز: القول إن "GenBank تحتوي على كل المعلومات الممكنة عن التسلسل". هذا غير صحيح؛ GenBank تحتوي على الكثير من المعلومات لكنها لا تحتوي على كل شيء (مثل البنية ثلاثية الأبعاد، درجات الجودة، التفاعلات الجزيئية). الصحيح هو: "GenBank هي الصيغة الأكثر غنى بين الصيغ الشائعة، وتحمل البيانات الوصفية والميزات والمراجع، لكنها لا تمثل النموذج الكامل للجزيء الحي." اختبار بسيط: فحص تطبيقي: "هل هذه الصيغة تحمل كل ما أحتاجه لتحليلي الحالي؟" وليس "هل تحمل كل شيء ممكن؟"ـ

Example: The presence of a sequence in a FASTA file with the identifier "BRCA1" is a computational representation. Claiming that "this sequence is the human BRCA1 gene" is a biological claim not proven by its presence in a file. The file tells us what its creator wrote, but it does not prove correctness. Practical test: before relying on an identity from a file, verify its source and confirm it with external references.

مثال: وجود تسلسل في ملف FASTA مع معرف "BRCA1" هو تمثيل حسابي. الادعاء بأن "هذا التسلسل هو جين BRCA1 البشري" هو ادعاء بيولوجي لا يثبته وجوده في ملف. الملف يخبرنا بما كتبه منشئه، لكنه لا يثبت صحته.

اختبار عملي: قبل أن تعتمد على هوية من ملف، تحقق من مصدرها وتأكد من صحتها بمصادر خارجية.

مثال تطبيقي على اختيار الصيغة:

· المدخلات: بيانات التسلسل، والسؤال العلمي، والأدوات التي ستستخدمها.

· الافتراضات: أن الصيغة المختارة تدعمها الأدوات التي ستستخدمها، وأنها تحمل المعلومات اللازمة لتحليلك.

· المخرجات: ملف بالصيغة المختارة، جاهز للاستخدام في الخطوة التالية.

· القيود: كل صيغة لها حدودها. FASTA لا تحمل جودة أو ميزات. FASTQ لا تحمل بيانات وصفية منظمة.

GenBank لا تحمل جودة وملفاتها أكبر.

Applied example of format selection:

اختبار: قبل اختيار صيغة، اكتب قائمة بالمعلومات التي تحتاجها لتحليلك، وتأكد من أن الصيغة تقدمها. إذا لم تقدمها،

فأنت بحاجة إلى صيغة أخرى أو إلى مصادر إضافية.

· Inputs: sequencing data, the scientific question, and the tools you will use.

· Assumptions: that the chosen format is supported by the project's tools, and that it carries the information needed for the project's analysis.

· Outputs: a file in the chosen format, ready for the next step.

· Limitations: each format has its limits. FASTA does not carry quality or features. FASTQ does not carry structured metadata. GenBank does not carry quality and files are larger.

Test: before choosing a format, write a list of the information the analysis requires for the project's analysis, and verify that the format provides it. If it does not, the analysis requires another format or additional sources.

عند العمل مع الصيغ، يجب تسجيل ما يلي:

· المصدر: الملف الأصلي (المسار، الاسم، التاريخ، المصدر).

· الإصدار: إصدار Biopython المستخدم، وإصدار الصيغة إذا كان معروفاً.

· المعلمات: الصيغة المستخدمة، وأي معاملات إضافية.

· التحويلات: أي تحويلات قمت بها بين الصيغ، مع توثيق ما فقد وما احتفظت به.

When working with formats, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا الملف وسجل التحويلات لزميل بعد عام، هل يستطيع فهم ما كان في الملف

الأصلي، وما تغير بعد التحويل؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the original file (path, name, date, source).

· Version: the Biopython version used, and the format version if known.

· Parameters: the format used, and any additional parameters.

· Transformations: any conversions you performed between formats, documenting what was lost and what was retained.

Test: Practical check: "If I gave this file and conversion log to a colleague a year later, could they understand what was in the original file, and what changed after conversion?" If the answer is no, the documentation is insufficient.

قبل أن تبدأ في تحليل بيانات حقيقية بصيغة معينة، أنشئ ملفات اصطناعية صغيرة بكل صيغة. أنشئ ملف FASTA، وملف FASTQ، وملف GenBank (إذا كان يوجد في المثال أداة لذلك)، واقرأها باستخدام SeqIO. تحقق من أن المعلومات التي تقرأها تطابق ما كتبته. ثم جرب التحويل بين الصيغ وراقب ما يُفقد. هذه الاختبارات تعلمك حدود كل صيغة قبل أن تواجهها في بيانات حقيقية كبيرة ومعقدة.

Before starting to analyze real data in a specific format, create small synthetic files in each format. Create a FASTA file, a FASTQ file, and a GenBank file (if the workflow has a tool for it), and read them using SeqIO. Verify that the information you read matches what you wrote. Then try converting between formats and observe what is lost. These tests teach you the limits of each format before you encounter them in large, complex real data.

Chapter 15 — Alignment, Similarity, and Homology

الفصل 15 — المحاذاة والتشابه والتماثل

Deeper Conceptual Explanation

الفكرة المركزية: alignment، identity، similarity، homology. النماذج والمرجع.

شرح أعمق للمفهوم

Alignment measures a computational relationship between sequences under a defined scoring scheme; it is not a final verdict about function. Alignment type, substitution matrix, and gap penalties affect the result. Homology is an evolutionary biological concept stronger than a similarity percentage and requires appropriate evolutionary and functional evidence.

المحاذاة طريقة لقياس علاقة حسابية بين تسلسلات وفق معيار محدد، وليست حكماً نهائياً على الوظيفة. اختيار نوع المحاذاة ومصفوفة الاستبدال والفجوات يؤثر في النتيجة. والتماثل التطوري مفهوم بيولوجي أقوى من مجرد نسبة تشابه، ويحتاج إلى قرائن تطورية ووظيفية مناسبة.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. What Is Alignment؟

1. ما هي المحاذاة (Alignment)؟

Alignment is the process of comparing two or more sequences by placing them side by side in a way that highlights similar and different regions. Imagine comparing two linguistic sentences: "the black cat" and "the dark cat." By aligning them, you see that "the cat" is shared, and "black" and "dark" are different but close in meaning. Alignment in sequences does the same: it places letters (nucleotides or amino acids) in parallel rows, and inserts gaps to improve matching. Alignment is a purely computational operation, relying on algorithms and predefined scores.

المحاذاة هي عملية مقارنة تسلسلين أو أكثر بوضعهم جنباً إلى جنب بطريقة تبرز المناطق المتشابهة والمختلفة. للتوضيح، يمكن تصور حالة يكون فيها تقارن جملتين لغويتين: "القطة السوداء" و "القط الأسود". بمحاذاتهما، ترى أن "القط" مشترك، وأن "السوداء" و "الأسود" مختلفان لكنهما قريبان في المعنى. المحاذاة في التسلسلات تفعل الشيء نفسه: تضع الحروف (القواعد النيتروجينية أو الأحماض الأمينية) في صفوف متوازية، وتُدخل فجوات (gaps) لتحسين التطابق. المحاذاة هي عملية حسابية بحتة، تعتمد على خوارزميات ودرجات محددة مسبقاً.

التسلسل 1: ATGCGTACGTAGCTAGCTAGCTAGCTAGC

التسلسل 2: ATGCGTACGTAGCTAGCTAGCTAGCT---GC

المحاذاة: ATGCGTACGTAGCTAGCTAGCTAGCTAGC

ATGCGTACGTAGCTAGCTAGCTAGCT---GC

Alignment does not tell you why these regions are similar; it simply arranges the letters to highlight the match. Interpreting this match is the next step, and this is where the other concepts come in.

2. Identity and Similarity

المحاذاة لا تخبرك لماذا هذه المناطق متشابهة؛ هي فقط ترتب الحروف بطريقة تبرز التطابق. تفسير هذا التطابق هو الخطوة التالية، وهنا تأتي المفاهيم الأخرى.

2. الهوية (Identity) والتشابه (Similarity)

Identity: is the percentage of positions where the letters match exactly between two sequences after alignment. If the workflow has 100 bases, and 80 of them match literally, the identity is 80%. Identity is a strict measure: either the letter matches or it does not.

الهوية (Identity) : هي النسبة المئوية للمواقع التي تتطابق فيها الحروف تماماً بين تسلسلين بعد المحاذاة. إذا كان يوجد في المثال 100 قاعدة، وتطابقت 80 منها حرفياً، فالهوية هي 80%. الهوية مقياس صارم: إما أن يتطابق الحرف أو لا.

Similarity: is the percentage of positions where the letters match exactly or are chemically similar (in proteins, for example, amino acids with similar properties such as polarity or size). Similarity is more tolerant than identity, because it takes into account conservative substitutions. For example, replacing an amino acid with another that has similar physicochemical properties may not change the protein's function.

Similarity (التشابه) : هي النسبة المئوية للمواقع التي تتطابق فيها الحروف تماماً أو تكون متشابهة كيميائياً (في البروتينات، مثلاً، الأحماض الأمينية ذات الخصائص المتشابهة مثل القطبية أو الحجم). التشابه أكثر تسامحاً من الهوية، لأنه يأخذ في الاعتبار البدائل المحافظة (conservative substitutions). مثلاً، استبدال حمض أميني بآخر له خصائص فيزيائية وكيميائية متشابهة قد لا يغير وظيفة البروتين.

```
# للبروتينات BLOSUM62 مثل) في الممارسة العملية، تستخدم مصفوفات الاستبدال
# لحساب التشابه، حيث تعطي المصفوفة درجة لكل زوج من الأحماض الأمينية
# بناءً على مدى تشابههما كيميائياً وتواترهما التطوري.
```

Identity is a number, and similarity is a number too. Both are computational outputs. But the most important question is: what does this number mean biologically?

3. Homology — The Evolutionary Hypothesis

الهوية هي رقم، والتشابه هو رقم أيضاً. كلاهما مخرجات حسابية. لكن السؤال الأهم هو: ماذا يعني هذا الرقم بيولوجياً؟

Homology is a biological hypothesis stating that two sequences (or genes, or proteins) share a common evolutionary ancestor. Homology is a relationship of descent: either the sequences are homologous (have a common ancestor) or they are not. There is no "percentage of homology"; homology is a binary property (yes/no). This is the essential difference between homology and similarity.

3. التماثل (Homology) — الفرضية التطورية

Similarity does not prove homology. Two sequences may be 80% similar because they share a common ancestor (homology), or because they evolved convergently (convergent evolution) to perform a similar function, or simply by chance. Similarity is computational evidence that may support a homology hypothesis, but it does not prove it. Proving homology requires additional evidence: phylogenetic analysis, presence of the same three-dimensional structure, functional evidence, and genetic context.

التماثل هو فرضية بيولوجية تقول إن تسلسلين (أو جينين، أو بروتينين) يشتركان في سلف تطوري مشترك. التماثل هو علاقة نسب: إما أن يكون التسلسلان متماثلين (لهما سلف مشترك) أو لا. لا يوجد "نسبة تماثل"؛ التماثل هو صفة ثنائية (نعم/لا). هذا هو الفرق الجوهري بين التماثل والتشابه.

التشابه لا يثبت التماثل. قد يكون تسلسلان متشابهين بنسبة 80% لأن لهما سلف مشترك (تماثل)، أو لأنهما تطورا بشكل متقارب (convergent evolution) لأداء وظيفة مماثلة، أو ببساطة بالصدفة. التشابه هو دليل حسابي قد يدعم فرضية التماثل، لكنه لا يثبتها. إثبات التماثل يحتاج إلى أدلة إضافية: تحليل أشجار التطور، ووجود نفس البنية الثلاثية، وأدلة وظيفية، وسياق جيني.

تشابه مرتفع (90%) + أدلة تطورية + بنية مشتركة = دليل قوي على التماثل.

تشابه مرتفع (90%) فقط = مجرد تشابه حسابي، قد يكون تماثلاً وقد لا يكون.

4. Why Is Similarity Not Enough؟

4. لماذا التشابه لا يكفي؟

A common mistake in published research is for the researcher to write: "The two sequences are 85% similar, therefore they are homologous." This conclusion is scientifically unjustified. Similarity is merely an indicator, and homology is a hypothesis that needs testing. For illustration, consider a case in which find two people who look alike, but they are not brothers. Similarity in appearance does not prove lineage. Similarly, sequence similarity does not prove common origin.

For this reason, the statement "similar" should always be specific: similar by what measure? (identity or similarity?). In which region? (the entire sequence or a specific part?). With which alignment? (algorithm type and gap parameters?). Under which model? (the substitution matrix used). This level of precision is what prevents a computational term from turning into an unjustified evolutionary conclusion.

من الأخطاء الشائعة في الأبحاث المنشورة أن يكتب الباحث: "التسلسلان متشابهان بنسبة 85%، لذلك هما متماثلان". هذا الاستنتاج غير مبرر علمياً. التشابه مجرد مؤشر، والتماثل فرضية تحتاج إلى اختبار. للتوضيح، يمكن تصور حالة يكون فيها تجد شخصين يشبهان بعضهما في الشكل، لكنهما ليسا أخوين. التشابه في الشكل لا يثبت النسب. وبالمثل، التشابه في التسلسل لا يثبت الأصل المشترك.

5. Practical Application: Using Alignment as a Tool

لهذا السبب، يجب أن تكون عبارة "متشابه" محددة دائماً: متشابه بأي مقياس؟ (الهوية أم التشابه؟). في أي منطقة؟ (التسلسل كاملاً أم جزء معين؟). وبأي محاذاة؟ (نوع الخوارزمية ومعايير الفجوات). وتحت أي نموذج؟ (مصفوفة الاستبدال المستخدمة). هذا المستوى من الدقة هو ما يمنع المصطلح الحسابي من التحول إلى استنتاج تطوري غير مبرر.

In practice, you will use alignment as an exploratory tool. You will align sequences, calculate similarity, and then ask: is this similarity high enough to generate a homology hypothesis? Then you will seek additional evidence to support or refute this hypothesis.

5. التطبيق العملي: استخدام المحاذاة كأداة

في التطبيق العملي، ستستخدم المحاذاة كأداة استكشافية. ستحاذي تسلسلات، وتحسب التشابه، ثم تسأل: هل هذا التشابه مرتفع بما يكفي لتوليد فرضية تماثل؟ ثم تبحث عن أدلة إضافية لدعم أو دحض هذه الفرضية.

```
# مثال توضيحي (غير تشغيلي) لكيفية التفكير
# لهما تشابه 95% حسب المحاذاة Y و X التسلسلان.
# الخطوة التالية: هل هذا التشابه موزع على كامل التسلسل أم في مناطق محددة؟
# هل هناك مناطق محفوظة وظيفياً؟
# هل للجينين نفس البنية الإكسونية-الإنترونية؟
# هل يتواجدان في نفس الموضع الكروموسومي في كائنات مختلفة؟
# كل هذه الأسئلة تنتقل بالتشابه من رقم إلى دليل.
```

6. Models and References

6. النماذج والمراجع

Alignment and similarity do not exist in a vacuum. They depend on models and references. Substitution matrices (such as BLOSUM62 and PAM250) are models that define the similarity score between amino acids based on their evolutionary frequency. Choosing the matrix affects the alignment and similarity results. The reference is the sequence you compare to. If you choose the wrong reference, you will get misleading similarity. Therefore, documenting the model and reference is an essential part of reproducibility.

المحاذاة والتشابه لا يوجدان في الفراغ. يعتمدان على النماذج والمراجع. مصفوفات الاستبدال (مثل BLOSUM62 و PAM250) هي نماذج تحدد درجة التشابه بين الأحماض الأمينية بناءً على تواترها التطوري. اختيار المصفوفة يؤثر على نتيجة المحاذاة والتشابه. المرجع هو التسلسل الذي تقارن به. إذا اخترت مرجعاً خاطئاً، فستحصل على تشابه مضلل. لذلك، توثيق النموذج والمرجع جزء أساسي من إعادة الإنتاج.

Chapter Summary

خلاصة الفصل

Alignment is a computational operation for comparing sequences. Identity and similarity are measurable outputs of this alignment. Homology is a biological hypothesis about common ancestry, and cannot be proven by similarity alone. Distinguishing between these three levels (computation, measurement, hypothesis) is the essence of critical thinking in bioinformatics. Using alignment as

a tool to generate hypotheses, and gathering additional evidence to test them, is the correct scientific method. The next chapter will take you into the world of biological algorithms, starting with exact and approximate sequence search.

المحاذاة هي عملية حسابية لمقارنة التسلسلات. الهوية والتشابه هما مخرجات قابلة للقياس من هذه المحاذاة. التماثل هو فرضية بيولوجية عن سلف مشترك، ولا يمكن إثباتها بالتشابه وحده. الفصل بين هذه المستويات الثلاثة (الحساب، القياس، الفرضية) هو جوهر التفكير النقدي في المعلوماتية الحيوية. استخدام المحاذاة كأداة لتوليد الفرضيات، وجمع أدلة إضافية لاختبارها، هو الطريقة العلمية السليمة. الفصل القادم سيدخلك إلى عالم الخوارزميات الحيوية، بدءاً بالبحث الدقيق والتقريبي في التسلسلات.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Alignment, Similarity, and Homology" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "المحاذاة والتشابه والتماثل" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "high similarity proves homology." This is incorrect; similarity is computational evidence that can support homology, but it does not prove it. The correct statement is: "high similarity is a strong indicator that may support a homology hypothesis, but proving homology requires additional evidence such as phylogenetic analysis, structure, and function." A simple test: if you find high similarity between two sequences, Practical check: "What other evidence supports the idea that they share a common ancestor?".

مثال على التجاوز: القول إن "التشابه العالي يثبت التماثل". هذا غير صحيح؛ التشابه دليل حسابي يمكن أن يدعم التماثل، لكنه لا يثبته. الصحيح هو: "التشابه العالي هو مؤشر قوي قد يدعم فرضية التماثل، لكن إثبات التماثل يتطلب أدلة إضافية مثل التحليل التطوري والبنية والوظيفة." اختبار بسيط: إذا وجدت تشابهاً عالياً بين تسلسلين، فحص تطبيقي: "ما الأدلة الأخرى التي تدعم فكرة أن لهما سلفاً مشتركاً؟".

Example: An alignment result showing 90% similarity is a computational representation. Claiming that "these two genes have the same function because they are 90% similar" is a biological claim not justified by similarity alone. Similarity tells us about letter matching, but it does not tell us about function in the cell. Practical test: before inferring function from similarity, look for independent evidence (such as gene knockout experiments, presence of the same functional domains, or expression context).

مثال: نتيجة محاذاة تعطي تشابهاً 90% هي تمثيل حسابي. الادعاء بأن "هذين الجينين لهما نفس الوظيفة لأنهما متشابهان 90%" هو ادعاء بيولوجي غير مبرر بالتشابه وحده. التشابه يخبرنا عن تطابق الحروف، لكنه لا يخبرنا عن الوظيفة في الخلية. اختبار عملي: قبل أن تستنتج وظيفة من التشابه، ابحث عن أدلة مستقلة (مثل تجارب تعطيل الجين، أو وجود نفس النطاقات الوظيفية، أو سياق التعبير).

مثال تطبيقي على المحاذاة باستخدام أداة مثل BLAST أو ClustalW:

· المدخلات: تسلسل استعلام (query)، وقاعدة بيانات مرجعية، ومصفوفة استبدال (مثل BLOSUM62 للبروتينات)، ومعاملات الفجوات.

· الافتراضات: أن المصفوفة المختارة مناسبة للمسافة التطورية، وأن معاملات الفجوات تعكس الواقع البيولوجي، وأن قاعدة البيانات المرجعية كاملة وصحيحة.

· المخرجات: محاذاة، ونسبة هوية/تشابه، ودرجة ثقة (مثل E-value في BLAST).

· القيود: المحاذاة لا تكتشف العلاقات البعيدة التي لا تظهرها المصفوفة، ولا تأخذ في الاعتبار البنية الثلاثية أو العوامل التنظيمية.

Applied example using an alignment tool such as BLAST or ClustalW:

اختبار: قبل أن تعتمد على نتيجة محاذاة، تحقق من المصفوفة والمعاملات المستخدمة، وفحص: "هل هذه الخيارات مناسبة لسؤالي البيولوجي؟".

· Inputs: a query sequence, a reference database, a substitution matrix (e.g., BLOSUM62 for proteins), and gap parameters.

· Assumptions: that the chosen matrix is appropriate for the evolutionary distance, that the gap parameters reflect biological reality, and that the reference database is complete and correct.

· Outputs: an alignment, percentage identity/similarity, and a confidence score (e.g., E-value in BLAST).

· Limitations: alignment does not detect distant relationships not captured by the matrix, nor does it account for three-dimensional structure or regulatory factors.

Test: before relying on an alignment result, check the matrix and parameters used, and ask: "Are these choices appropriate for my biological question?".

عند إجراء محاذاة، يجب تسجيل ما يلي:

· المصدر: التسلسلات المستخدمة (معرفاتها ومصادرها).

· الإصدار: إصدار الأداة المستخدمة (مثل BLAST 2.12.0، أو ClustalW 2.1)، وإصدار قاعدة البيانات المرجعية.

· المعلمات: مصفوفة الاستبدال، ومعاملات الفجوات، وأي عتبات (مثل E-value).

· التحويلات: أي معالجة مسبقة للتسلسلات قبل المحاذاة (مثل قص المناطق منخفضة الجودة).

When performing an alignment, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة نفس المحاذاة والحصول على نفس النتائج باستخدام نفس المعلمات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the sequences used (their identifiers and sources).

· Version: the tool version used (e.g., BLAST 2.12.0, or ClustalW 2.1), and the reference database version.

· Parameters: the substitution matrix, gap parameters, and any thresholds (e.g., E-value).

· Transformations: any preprocessing of sequences before alignment (e.g., trimming low-quality regions).

Test: Practical check: "If I gave this record to a colleague a year later, could they reproduce the same alignment and obtain the same results using the same parameters?" If the answer is no, the documentation is insufficient.

قبل أن تجري محاذاة على تسلسلات حقيقية، اختبر أداة المحاذاة على تسلسلات اصطناعية ذات تشابه معروف. أنشئ تسلسلين لهما 100% هوية، وتأكد من أن الأداة تعطي هذا التشابه. ثم أنشئ تسلسلين لهما تشابه 70% مع إدخال فجوات، وتأكد من أن المحاذاة تضع الفجوات في المكان الصحيح. إذا نجحت هذه الاختبارات، فأنت واثق من أن الأداة تعمل كما هو متوقع، وأي نتيجة على بيانات حقيقية تعكس العلاقات الحقيقية وليس خطأ في الإعدادات.

Before performing an alignment on real sequences, test the alignment tool on synthetic sequences with known similarity. Create two sequences with 100% identity, and verify that the tool gives this similarity. Then create two sequences with 70% similarity with inserted gaps, and verify that the alignment places the gaps correctly. If these tests succeed, the workflow is confident that the tool works as expected, and any result on real data reflects true relationships and not an error in the settings.

Part 5 — Supporting Tools

الباب 5 — الأدوات المساندة

الفكرة المركزية: pipeline thinking، الملفات، التحويلات، السجل، العزل.

Chapter 16 — Bash and the Command Line

الفصل 16 — Bash وسطر الأوامر

شرح أعمق للمفهوم

Deeper Conceptual Explanation

Bash يربط الأدوات أكثر مما يحل محلها. قوته في تمرير المخرجات بين الخطوات، وإدارة الملفات، وتسجيل الأوامر، وبناء pipeline يمكن إعادة تشغيله. لكن السلسلة الطويلة من الأوامر تحتاج إلى فحص المدخلات والمخرجات وحالات الفشل، وإلا تحولت سهولة التشغيل إلى صعوبة في تتبع الخطأ.

Bash is valuable for connecting tools rather than replacing them. Its strength lies in passing outputs between stages, managing files, recording commands, and building rerunnable pipelines. Long command chains still require input, output, and failure checks; otherwise convenience becomes difficulty when tracing errors.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Why Bash؟

1. لماذا Bash؟

Bash is the default command-line language on most Linux and macOS systems, and is also available on Windows through WSL or Git Bash. It is the language you use to run tools, manage files, and connect inputs to outputs. In bioinformatics, many tools (such as BLAST, FastQC, Samtools) run from the command line, not from a graphical interface. Learning Bash gives you the ability to run these tools efficiently, assemble them into pipelines, and automate repetitive tasks.

Bash هي لغة سطر الأوامر الافتراضية في معظم أنظمة Linux و macOS، ومتاحة أيضاً على Windows من خلال WSL أو Git Bash. هي اللغة التي تستخدمها لتشغيل الأدوات، وإدارة الملفات، وربط المدخلات بالمخرجات. في المعلوماتية الحيوية، العديد من الأدوات (مثل BLAST، FastQC، Samtools) تعمل من سطر الأوامر، وليس من واجهة رسومية. تعلم Bash يمنحك القدرة على تشغيل هذه الأدوات بكفاءة، وتجميعها في خطوط أنابيب، وأتمتة المهام المتكررة.

But more important than the commands themselves is pipeline thinking. Instead of thinking of each step in isolation, you think of a series of transformations: a file enters, a tool processes it, produces another file, which enters the next tool, and so

on. This thinking is what makes analysis traceable, documentable, and reproducible.

لكن الأهم من الأوامر نفسها هو التفكير في خطوط الأنابيب (pipeline thinking) . بدلاً من التفكير في كل خطوة على حدة، تفكر في سلسلة من التحويلات: ملف يدخل، تمر عليه أداة، تنتج ملفاً آخر، يدخل إلى الأداة التالية، وهكذا. هذا التفكير هو ما يجعل التحليل قابلاً للتتبع، والتوثيق، وإعادة الإنتاج.

2. Basic Command-Line Concepts

2. المفاهيم الأساسية لسطر الأوامر

المسار (Path) : هو عنوان الملف أو المجلد في نظام التشغيل. المسار النسبي يبدأ من موقعك الحالي (مثل data/sequences.fasta)، والمسار المطلق يبدأ من الجذر (مثل /home/user/projects/data/sequences.fasta). فهم المسارات ضروري لتحديد المدخلات والمخرجات بدقة.

الملفات والمجلدات: تنظيم الملفات في مجلدات هو جزء من تصميم سير العمل. مجلد للمدخلات، ومجلد للنتائج الوسيطة، ومجلد للمخرجات النهائية. هذا التنظيم يمنع الفوضى ويجعل كل خطوة قابلة للتتبع.

الأوامر الأساسية:

- ls: عرض محتويات المجلد.
- cd: تغيير المجلد الحالي.
- mkdir: إنشاء مجلد جديد.
- cp: نسخ ملف أو مجلد.
- mv: نقل أو إعادة تسمية ملف.

Path: is the address of a file or folder in the operating system. A relative path starts from the project's current location (e.g., data/sequences.fasta), while an absolute path starts from the root (e.g., /home/user/projects/data/sequences.fasta). Understanding paths is essential for precisely specifying inputs and outputs.

- rm: حذف ملف (استخدم بحذر شديد).

Files and Directories: Organizing files into directories is part of workflow design. A directory for inputs, one for intermediate results, and one for final outputs. This organization prevents chaos and makes every step traceable.

- head / tail: عرض بداية أو نهاية ملف نصي.

Basic Commands:

· ls: list directory contents.

· cd: change directory.

· mkdir: create a new directory.

· cp: copy a file or directory.

· mv: move or rename a file.

· rm: delete a file (use with extreme caution).

· head / tail: display the beginning or end of a text file.

Language / نوع الكود: **Bash / Shell**

```
# مثال: إنشاء مجلدات وتنظيم الملفات
mkdir data results logs
```

```
cp original_sequences.fasta data/
```

```
cd data
```

```
ls
```

```
head sequences.fasta
```

3. Pipelines

A pipeline is connecting several commands so that the output of the first command becomes the input of the second, and so on. In Bash, the | (pipe) symbol is used to connect commands. This allows data processing without intermediate files, making the process more efficient and easier to document.

3. خطوط الأنابيب (Pipelines)

خط الأنابيب هو ربط عدة أوامر معاً بحيث يمر مخرج الأمر الأول إلى مدخل الأمر الثاني، وهكذا. في Bash، يُستخدم الرمز | (الأنبوب) لربط الأوامر. هذا يسمح بمعالجة البيانات دون الحاجة إلى ملفات وسيطة، مما يجعل العملية أكثر كفاءة وأسهل في التوثيق.

Language / نوع الكود: **Bash / Shell**

Purpose: perform command-line text processing as a step that can be composed into a pipeline.

الغرض: معالجة نصية من سطر الأوامر ضمن خطوة يمكن دمجها في pipeline.

```
# مثال: عرض أول 10 أسطر من ملف، ثم البحث عن كلمة "ATG"
```

```
head sequences.fasta | grep "ATG"
```

```
# مثال: عد عدد التسلسلات في ملف FASTA (عندما يكون كل تسلسل في سطر واحد)
grep "^>" sequences.fasta | wc -l
```

في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. أوامر الطرفية حساسة للسياق: نظام التشغيل، PATH، مجلد العمل، صلاحيات الملفات، وإصدارات الأدوات. لذلك يجب تسجيل الأمر كما نُفذ، مع مدخلاته ومخرجاته، بدل حفظ النتيجة وحدها.

But in bioinformatics, the tools you use often work on files rather than directly on text. Therefore, you often use intermediate files, and document each step. The principle is the same: each step has a documented input and output.

4. Logging

لكن في المعلوماتية الحيوية، غالباً ما تكون الأدوات التي تستخدمها تعمل على ملفات وليس على نص مباشر. لذلك، غالباً ما تستخدم ملفات وسيطة، وتوثق كل خطوة. المبدأ هو نفسه: كل خطوة لها مدخل ومخرج موثق.

A command log is a file that records every command you ran, its date, and its outputs. In Bash, the workflow can use the script command to record an entire session, or the workflow can write commands in a text file (a script) and run it. A good log allows you and others to know exactly what happened during the analysis. This is the essence of reproducibility.

4. سجل الأوامر (Logging)

سجل الأوامر هو ملف يسجل كل أمر شغّلته، وتاريخه، ومخرجاته. في Bash، يمكن استخدام أمر script لتسجيل جلسة كاملة، أو يمكن كتابة الأوامر في ملف نصي (سكربت) وتشغيله. السجل الجيد يسمح لك وللآخرين بمعرفة بالضبط ما حدث أثناء التحليل. هذا هو جوهر إعادة الإنتاج.

Language / نوع الكود: **Bash / Shell**

Purpose: connect sequential analysis tools in a workflow with explicit inputs, outputs, and logs.

الغرض: ربط أدوات تحليل متتابعة في سير عمل، مع فصل المدخلات والمخرجات والسجلات.

```
# تشغيل الأوامر
fastqc raw_data.fastq
```

```
cutadapt -a AGATCGGAAGAGC ... -o trimmed.fastq raw_data.fastq

bowtie2 -x reference -U trimmed.fastq -S aligned.sam
```

```
# بدء تسجيل جلسة
script analysis_log.txt
```

```
# إنهاء التسجيل
exit
```

5. Organizing Inputs and Outputs

5. تنظيم المدخلات والمخرجات

من الأخطاء الشائعة أن تعمل على ملفات في مجلدات غير منظمة، أو أن تستخدم أسماء ملفات عامة (مثل output.txt). هذا يجعل من الصعب تتبع ما أنتجه كل خطوة. القاعدة الأساسية هي:

· كل خطوة لها مجلد خاص بها (أو اسم ملف مميز).

· أسماء الملفات تحمل معلومات عن المحتوى والتاريخ.

A common mistake is to work with files in disorganized directories, or to use generic file names (like output.txt). This makes it difficult to track what each step produced. The basic rule is:

· المدخلات والمخرجات مفصولة في مجلدات مختلفة.

· Each step has its own directory (or a distinct file name).

· لا تعمل على الملفات الأصلية مباشرة؛ احتفظ بنسخة منها في مجلد raw.

· File names carry information about content and date.

· Inputs and outputs are separated into different directories.

· Do not work on original files directly; keep a copy in a raw directory.

```
# هيكل مجلدات مقترح
project/
├── raw/          # البيانات الخام (لا تتغير)
├── processed/    # البيانات بعد التنقية
├── analysis/     # نتائج التحليل
├── logs/         # سجلات الأوامر
└── scripts/      # سكربتات Bash
```

6. العزل (Isolation)

6. Isolation

في الفصول السابقة، تعلمت عن البيئات الافتراضية لعزل اعتماديات Python. في Bash، العزل يعني تشغيل أدوات في بيئات منفصلة لتجنب تعارض الإصدارات. يمكن استخدام أدوات مثل Conda أو Docker لعزل الأدوات

والاعتماديات. الفكرة هي أن كل مشروع يجب أن يكون لديه بيئته الخاصة، سواء كانت بايثون، أو R، أو أدوات Bash.

هذا يمنع تعارض الإصدارات ويضمن إعادة الإنتاج.

Language / نوع الكود: **Bash / Shell**

```
# استخدام Conda لإنشاء بيئة معزولة لأدوات معينة
conda create -n my_project python=3.11
conda activate my_project
conda install -c bioconda fastqc cutadapt bowtie2
```

7. Practical Application: Building a Simple Pipeline

Let us build a simple pipeline for sequence data analysis:

7. التطبيق العملي: بناء خط أنابيب بسيط

دعنا نبني خط أنابيب بسيطاً لتحليل بيانات تسلسل:

Language / نوع الكود: **Bash / Shell**

```
#!/bin/bash
# pipeline.sh - خط أنابيب بسيط لتحليل FASTQ
# 2. تنقية البيانات
cutadapt -a AGATCGGAAGAGC -o data/processed/trimmed.fastq \
  data/raw/sample.fastq \
  2> logs/cutadapt.log

# 1. إنشاء المجلدات
mkdir -p data/raw data/processed results logs
# 3. فحص الجودة
fastqc data/processed/trimmed.fastq -o results/ \
  2> logs/fastqc.log

# 4. محاذاة التسلسلات (مثال باستخدام bowtie2)
bowtie2 -x reference/index -U data/processed/trimmed.fastq -S \
  results/aligned.sam \
  2> logs/bowtie2.log

# 5. عد القراءات
samtools view -c results/aligned.sam > results/read_count.txt

echo "اكتمل التحليل في $(date)" >> logs/pipeline.log
```

This script can be run with a single command, documents every step, separates inputs from outputs, and logs errors. This is the essence of pipeline thinking: making analysis repeatable and documentable.

هذا السكربت يمكن تشغيله بأمر واحد، وهو يوثق كل خطوة، ويفصل المدخلات عن المخرجات، ويسجل الأخطاء. هذا هو جوهر pipeline thinking: جعل التحليل قابلاً للتكرار والتوثيق.

Chapter Summary

خلاصة الفصل

Bash and the command line are the fundamental tools for organizing workflows in bioinformatics. Pipeline thinking is thinking of analysis as a series of traceable transformations. Organizing files into directories, using distinct names, logging commands, and isolating environments are all practices that make analysis reproducible. Bash is not just a language for running commands; it is a way of organizing scientific thinking itself. The next chapter will present Perl as a historical and optional tool for text processing in bioinformatics.

Bash وسطر الأوامر هما الأدوات الأساسية لتنظيم سير العمل في المعلوماتية الحيوية. pipeline thinking هو التفكير في التحليل كسلسلة من التحويلات القابلة للتتبع. تنظيم الملفات في مجلدات، واستخدام أسماء مميزة، وتسجيل الأوامر، وعزل البيئات كلها ممارسات تجعل التحليل قابلاً لإعادة الإنتاج. Bash ليست مجرد لغة لتشغيل الأوامر؛ هي طريقة لتنظيم التفكير العلمي نفسه. الفصل القادم سيعرض لك Perl كأداة تاريخية واختيارية لمعالجة النصوص في المعلوماتية الحيوية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Bash and the Command Line" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Bash وسطر الأوامر" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "using Bash guarantees the correctness of biological results." This is incorrect; Bash organizes workflows and facilitates reproducibility, but it does not verify biological hypotheses. The correct statement is: "Bash provides a way to organize and document analysis steps, making it easier to trace errors and reproduce results, but the validity of biological results depends on data quality and sound analysis." A simple test: Practical check: "Does my Bash script document every step, or does it act as a black box?".

مثال على التجاوز: القول إن "استخدام Bash يضمن صحة النتائج البيولوجية". هذا غير صحيح؛ Bash تنظم سير العمل وتسهل إعادة الإنتاج، لكنها لا تتحقق من صحة الفرضيات البيولوجية. الصحيح هو: "Bash توفر طريقة لتنظيم وتوثيق خطوات التحليل، مما يسهل تتبع الأخطاء وإعادة الإنتاج، لكن صحة النتائج البيولوجية تعتمد على جودة البيانات وسلامة التحليل." اختبار بسيط: فحص تطبيقي: "هل سكربت Bash الخاص بي يوثق كل خطوة، أم أنه يعمل كصندوق أسود؟"ـ

Example: Running a Bash script that analyzes data and produces numerical results is a computational representation. Claiming that "these results prove a biological relationship" is a biological claim not proven by merely running a script. The script executes specific steps, but interpreting the results biologically requires context and additional knowledge. Practical test: before interpreting the results of a Bash script, Practical check: "Is every step in the script documented and understood? Are the results consistent with existing biological knowledge?".

مثال: تشغيل سكربت Bash يحلل بيانات ويعطي نتائج رقمية هو تمثيل حسابي. الادعاء بأن "هذه النتائج تثبت وجود علاقة بيولوجية" هو ادعاء بيولوجي لا يثبته مجرد تشغيل سكربت. السكربت ينفذ خطوات محددة، لكن تفسير النتائج بيولوجياً يحتاج إلى سياق ومعرفة إضافية. اختبار عملي: قبل أن تفسر نتائج سكربت Bash، فحص تطبيقي: "هل كل خطوة في السكربت موثقة ومفهومة؟ وهل النتائج متسقة مع المعرفة البيولوجية الموجودة؟"ـ

مثال تطبيقي على سكربت Bash لتحليل FASTQ:

· المدخلات: ملف FASTQ خام، ملف مرجعي للمحاذاة، معاملات الأدواتـ

· الافتراضات: أن الأدوات مثبتة بشكل صحيح، وأن المسارات صحيحة، وأن الملفات المدخلة سليمةـ

· المخرجات: ملفات معالجة (تنقية، محاذاة)، تقارير جودة، سجلاتـ

· القيود: السكربت لا يتحقق من صحة المدخلات قبل التشغيل، ولا يتعامل مع الأخطاء بشكل شامل، ولا يوثق إصدارات الأدواتـ

Applied example of a Bash script for FASTQ analysis:

اختبار: قبل تشغيل سكربت على بيانات حقيقية، اختبره على ملف اختبار صغير، وتأكد من أن المخرجات متوقعةـ

· Inputs: raw FASTQ file, reference file for alignment, tool parameters.

· Assumptions: that the tools are installed correctly, that paths are correct, and that the input files are intact.

· Outputs: processed files (trimming, alignment), quality reports, logs.

· Limitations: the script does not validate inputs before running, nor handle errors comprehensively, nor document tool versions.

Test: before running the script on real data, test it on a small test file, and verify that the outputs are as expected.

عند كتابة سكربت Bash، يجب تسجيل ما يلي:

· المصدر: الملفات الأصلية ومواقعهاـ

· الإصدار: إصدارات الأدوات المستخدمة (يمكن تسجيلها باستخدام --version مع كل أداة).

· المعلمات: جميع المعلمات المستخدمة لكل أداة.

· التحويلات: تسلسل الخطوات التي يطبقها السكربت.

When writing a Bash script, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السكربت وزملائي بعد عام، هل يستطيعون تشغيله والحصول على نفس النتائج دون تعديلات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: original files and their locations.

· Version: versions of the tools used (can be recorded using --version with each tool).

· Parameters: all parameters used for each tool.

· Transformations: the sequence of steps the script applies.

Test: Practical check: "If I gave this script and colleagues a year later, could they run it and get the same results without modifications?" If the answer is no, the documentation is insufficient.

قبل تشغيل سكربت Bash على بيانات حقيقية، اختبره على بيانات اصطناعية صغيرة. أنشئ ملف FASTQ صغيراً يحتوي على بضع قراءات ذات تسلسلات معروفة، وشغّل السكربت عليه. تحقق من أن المخرجات (التنقية، المحاذاة، التقارير) تتطابق مع التوقعات. إذا نجح الاختبار، فأنت واثق من أن السكربت يعمل كما هو متوقع، وأي مشكلة مع بيانات حقيقية تعكس جودة البيانات وليس خطأ في السكربت.

Before running a Bash script on real data, test it on small synthetic data. Create a small FASTQ file containing a few reads with known sequences, and run the script on it. Verify that the outputs (trimming, alignment, reports) match expectations. If the test succeeds, the workflow is confident that the script works as expected, and any problem with real data reflects data quality, not an error in the script.

Chapter 17 — Perl in a Limited Dose

الفصل 17 — Perl بجرعة محددة

الفكرة المركزية: regex، legacy scripts، صيانة الأدوات، افتراضات الملفات.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

يظهر Perl هنا كأداة لمعالجة النصوص والملفات والأنماط، لا كبديل عن Python أو Biopython. قوته التاريخية في النصوص والتعبيرات النمطية، لكن جودة السكربت تعتمد على وضوح الافتراضات وتعاملها مع الملفات متعددة الأسطر وحالات الإدخال غير المتوقعة.

Perl appears here as a tool for text and file processing rather than a replacement for Python or Biopython. Its historical strength lies in text manipulation and regular expressions, but a reliable script still depends on explicit assumptions and correct handling of multiline files and unexpected input.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Why Perl؟

1. لماذا Perl؟

Perl was the preferred language for bioinformatics in the 1990s and early 2000s for several reasons. It was very powerful for text processing and regular expressions, which is exactly what the analysis requires when dealing with sequence files and various formats. It was fast for writing short scripts that perform complex tasks. And many core databases and tools were written in it. Even today, some older tools still use Perl, and you may encounter Perl scripts in legacy projects or in laboratories that rely on stable systems. You do not need to become an expert in Perl, but the analysis requires to be able to read it and modify it when necessary, and understand what it does before relying on it.

كانت Perl اللغة المفضلة للمعلوماتية الحيوية في التسعينيات وأوائل الألفينيات لعدة أسباب. كانت قوية جداً في معالجة النصوص والتعبيرات النمطية، وهو ما تحتاجه كثيراً عند التعامل مع ملفات التسلسل والصيغ المختلفة. كانت سريعة في كتابة سكربتات قصيرة تؤدي مهام معقدة. والعديد من قواعد البيانات والأدوات الأساسية كُتبت بها. حتى اليوم، بعض الأدوات القديمة لا تزال تستخدم Perl، وقد تجد سكربتات Perl في مشاريع قديمة أو في مختبرات تعتمد على أنظمة مستقرة. لا تحتاج إلى أن تصبح خبيراً في Perl، لكنك بحاجة إلى أن تكون قادراً على قراءتها وتعديلها عند الضرورة، وأن تفهم ما تفعله قبل أن تعتمد عليها.

Purpose: process text or read a file in Perl while keeping regular expressions and assumptions explicit.

الغرض: معالجة نصية أو قراءة ملف باستخدام Perl، مع إبقاء التعبيرات النمطية والافتراضات واضحة.

```
my $count = 0;
```

```
while (<>) {
  if (/^>/) {
```

```
$count++;
}
}
print "Total sequences: $count\n";
```

```
# مثال بسيط لسكربت Perl يقرأ ملف FASTA ويحسب عدد التسلسلات
#!/usr/bin/perl
```

```
use strict;
use warnings;
```

In Perl, strict and warnings are not cosmetic additions; they reduce hidden state and errors that can pass unnoticed in short scripts. Textual biological data should also be checked for line structure, field counts, and encoding before computation proceeds.

في Perl، تفعيل strict وwarnings ليس تحسينًا شكليًا؛ إنه يقلل من الحالة الخفية والأخطاء التي قد تمر في نصوص قصيرة. عند معالجة بيانات حيوية نصية يجب أيضًا التحقق من شكل السطر وعدد الحقول والترميز قبل الانتقال إلى الحساب.

This example is used for its focused text or file-processing role. In a real workflow, pin the environment, validate inputs, and handle read/write failures explicitly.

يُستخدم هذا المثال في موضعه المحدد لمعالجة النصوص أو الملفات. عند إدخاله في خط عمل حقيقي يجب تثبيت الإصدار، والتحقق من المدخلات، والتعامل الصريح مع أخطاء القراءة والكتابة.

2. Regular Expressions

هذا السكربت البسيط يقرأ ملفاً (أو عدة ملفات) من سطر الأوامر، ويبحث عن الأسطر التي تبدأ بعلامة > (وهي بداية كل تسلسل في FASTA)، ويعدها، ثم يطبع العدد.

Regular expressions are a language for searching for patterns within text. They are one of Perl's most powerful features, and are also used in Python and other tools. In Perl, regular expressions are written between forward slashes (/pattern/). the workflow can use them to search for specific sequences, extract parts of text, or verify file formats.

2. التعبيرات النمطية (Regular Expressions)

التعبيرات النمطية هي لغة للبحث عن أنماط داخل النصوص. هي واحدة من أقوى ميزات Perl، وتُستخدم أيضاً في Python وأدوات أخرى. في سياق Perl، التعبيرات النمطية مكتوبة بين شرطتين مائلتين (/pattern/). يمكن استخدامها للبحث عن تسلسلات معينة، أو استخراج أجزاء من النص، أو التحقق من تنسيق الملفات.

```
# أمثلة على تعبيرات نمطية في Perl
# البحث عن تسلسل يحتوي على 'ATG' (كودون بدء)
if ($seq =~ /ATG/) {
    print "Found start codon\n";
}

# استخراج أرقام من سطر
if ($line =~ /(\d+)/) {
    my $number = $1;
    print "Found number: $number\n";
}

# التحقق من أن السطر هو رأس FASTA
if ($line =~ /^>/) {
    print "This is a FASTA header\n";
}
```

Understanding regular expressions is important not only for reading Perl, but also for using them in Python (re module), and in command-line tools like grep and sed. They are a fundamental skill for text processing in bioinformatics.

3. Risks of Maintaining Legacy Tools

فهم التعبيرات النمطية مهم ليس فقط لقراءة Perl، ولكن أيضاً لاستخدامها في Python (re module)، وفي أدوات سطر الأوامر مثل grep و sed. هي مهارة أساسية لمعالجة النصوص في المعلوماتية الحيوية.

One of the biggest challenges in dealing with legacy Perl scripts is that their assumptions about file formats may not be documented. The script might assume the file has one line per sequence, or that the identifier follows a specific pattern, or that the file has no empty lines. When these assumptions change (e.g., if the file becomes multi-line sequences, or if the identifier format changes), the script may fail or silently produce incorrect results. Maintaining legacy tools requires understanding these assumptions, documenting them, and updating them when necessary.

3. مخاطر صيانة الأدوات القديمة

أحد أكبر التحديات في التعامل مع سكربتات Perl القديمة هو أن افتراضاتها حول تنسيق الملفات قد لا تكون موثقة. قد يفترض السكربت أن الملف يحتوي على سطر واحد لكل تسلسل، أو أن المعرف يتبع نمطاً معيناً، أو أن الملف خالٍ

من الأسطر الفارغة. عندما تتغير هذه الافتراضات (مثلاً، إذا أصبح الملف يحتوي على تسلسلات متعددة الأسطر، أو إذا تغير تنسيق المعرف)، قد يفشل السكربت أو يعطي نتائج خاطئة بصمت. صيانة الأدوات القديمة تتطلب فهم هذه الافتراضات، وتوثيقها، وتحديثها عند الحاجة.

```
# سكربت Perl قديم يعتمد على افتراض أن كل تسلسل في سطر واحد
# إذا كان الملف يحتوي على تسلسلات متعددة الأسطر، فإن هذا السكربت سيفشل
while (<>) {
    chomp;
    if (/^>/) {
        print "$_\n";  # طباعة الرأس
    } else {
        print "$_\n";  # طباعة التسلسل (يفترض أنه سطر واحد)
    }
}
```

4. Reading Legacy Tools as a Lesson in Documentation

هذا السكربت البسيط يفترض أن التسلسل كله في سطر واحد. إذا كان الملف يحتوي على تسلسل مقسم على عدة أسطر، فلن يعمل بشكل صحيح. هذا هو مثال على افتراض غير موثق يمكن أن يسبب مشاكل.

Reading legacy Perl scripts teaches you an important lesson: the script may depend on a file format that is no longer clear after years. The script may have been written at a time when the format was different, or when standards were less strict. When you read a legacy script, it is appropriate to ask: what are this script's assumptions? What inputs does it expect? What are the outputs? Is there documentation describing this? This lesson applies to any code you write today: document the project’s assumptions, write a description of inputs and outputs, and make the code understandable to others (and to the analysis itself a year later).

4. قراءة الأدوات القديمة كدرس في التوثيق

قراءة سكربتات Perl القديمة تعلمك درساً مهماً: النص البرمجي قد يعتمد على شكل ملف لم يعد واضحاً بعد سنوات. قد يكون السكربت كُتب في زمن كانت فيه الصيغة مختلفة، أو كانت المعايير أقل صرامة. عندما تقرأ سكربتاً قديماً، يجب أن تسأل: ما هي افتراضات هذا السكربت؟ ما هي المدخلات التي يتوقعها؟ ما هي المخرجات؟ وهل هناك وثائق تصف ذلك؟ هذا الدرس ينطبق على أي كود تكتبه اليوم: وثّق افتراضاتك، واكتب وصفاً للمدخلات والمخرجات، واجعل الكود مفهوماً للآخرين (ولنفسك بعد عام).

```
# مثال على سكربت مع توثيق جيد
#!/usr/bin/perl
```

```
# count_sequences.pl - يحسب عدد التسلسلات في ملف FASTA
# المدخلات: ملف FASTA (يمكن أن يكون متعدد الأسطر)
# المخرجات: عدد التسلسلات
# الافتراضات: الرأس يبدأ بـ '>'، ويتكون التسلسل من أحرف متصلة
# الإصدار: 1.0 (15-01-2020)
# المؤلف: user@example.com
```

5. The Limited Dose in This Book

5. الجرعة المحددة في هذا الكتاب

في هذا الكتاب، لا نتعلم Perl كلغة برمجة متكاملة. نتعلم ما يكفي لقراءة وفهم السكربتات القديمة. نتعلم:

· هيكل السكربت الأساسي (السطر الأول، use strict؛ use warnings؛).

· المتغيرات الأساسية (المقاييس، المصفوفات، القواميس).

· التعبيرات النمطية الأساسية.

· قراءة الملفات سطراً بسطر.

In this book, we do not learn Perl as a complete programming language. We learn enough to read and understand legacy scripts. We learn:

· كتابة مخرجات بسيطة.

· The basic script structure (shebang line, use strict; use warnings;).

إذا احتجت إلى كتابة كود جديد لمعالجة النصوص، استخدم Python بدلاً من Perl. لكن إذا واجهت سكربتاً قديماً، ستكون قادراً على فهم ما يفعله، وتعديله إذا لزم الأمر، وتوثيق افتراضاته. هذه الجرعة المحددة كافية للتعامل مع الإرث التاريخي للمعلوماتية الحيوية.

· Basic variables (scalars, arrays, hashes).

· Basic regular expressions.

· Reading files line by line.

· Writing simple outputs.

If the analysis requires to write new code for text processing, use Python instead of Perl. But if you encounter a legacy script, you will be able to understand what it does, modify it if necessary, and document its assumptions. This limited dose is sufficient for dealing with bioinformatics' historical legacy.

6. Practical Application: Reading a Legacy Perl Script

Suppose you find the following script in a legacy project:

6. التطبيق العملي: قراءة سكربت Perl قديم

لنفترض أنك وجدت السكربت التالي في مشروع قديم:

```
#!/usr/bin/perl
while (<>) {
    chomp;
    if (/^>(\S+)/) {
        print "ID: $1\n";
    }
}
```

تُقرأ هذه الكتلة بوصفها خطوة محددة داخل سير العمل: ما المدخل؟ ما التحويل؟ وما المخرج الذي يمكن اختباره؟ هذه القراءة تمنع فصل الكود عن السياق العلمي الذي يعطيه معناه.

هذا السكربت يقرأ ملفاً، ويتجاهل كل الأسطر باستثناء رؤوس FASTA، ويطبع المعرف الموجود بعد علامة > وحتى أول مسافة. السؤال: ما هي افتراضات هذا السكربت؟

· يفترض أن رأس FASTA يبدأ بعلامة >.

· يفترض أن المعرف هو أول كلمة بعد > (يتوقف عند أول مسافة).

· يفترض أن الملف ليس كبيراً جداً (لأنه يقرأ سطراً سطراً بدون مشكلة).

This script reads a file, ignores all lines except FASTA headers, and prints the identifier after the > up to the first space. The question: what are this script's assumptions?

· لا يفترض أي شيء عن التسلسلات نفسها (يتجاهلها تماماً).

· It assumes that the FASTA header starts with >.

إذا كان تنسيق رأس FASTA في ملفك يختلف (مثلاً، إذا كان المعرف يحتوي على مسافات)، فقد لا يعمل السكربت بشكل صحيح. هذا هو المثال العملي لفهم افتراضات الأدوات القديمة.

· It assumes that the identifier is the first word after > (stops at the first space).

· It assumes the file is not too large (since it reads line by line without issue).

· It assumes nothing about the sequences themselves (it ignores them entirely).

If the FASTA header format in the project's file is different (e.g., if the identifier contains spaces), the script may not work correctly. This is a practical example of understanding legacy tool assumptions.

Chapter Summary

خلاصة الفصل

Perl is a historical language with a strong presence in bioinformatics, and many tools and legacy scripts were written in it. In this chapter, you learned to read and understand these scripts, recognize regular expressions, and become aware of the

risks of undocumented assumptions about file formats. The goal is not to learn Perl as a new programming language, but to gain the skill of dealing with historical legacy. The next chapter will introduce you to R as a tool for statistics and visualization in bioinformatics.

Perl هي لغة تاريخية ذات حضور قوي في المعلوماتية الحيوية، والعديد من الأدوات والسكربتات القديمة كُتبت بها. في هذا الفصل، تعلمت قراءة وفهم هذه السكربتات، والتعرف على التعبيرات النمطية، وإدراك مخاطر الافتراضات غير الموثقة حول تنسيق الملفات. الهدف ليس تعلم Perl كلغة برمجة جديدة، بل اكتساب مهارة التعامل مع الإرث التاريخي. الفصل القادم سيعرض لك R كأداة للإحصاء والتصور في المعلوماتية الحيوية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Perl in a Limited Dose" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Perl بجرعة محددة" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "Perl is the best language for text processing in bioinformatics." This is incorrect; Perl was the best in the past, but Python has become the more widely used choice. The correct statement is: "Perl was a foundational language in bioinformatics, and many legacy tools are still written in it, so understanding it is useful for reading legacy scripts, but for new projects Python is preferred." A simple test: Practical check: "Do I need to write new code, or just read legacy code?".

مثال على التجاوز: القول إن "Perl هي اللغة الأفضل لمعالجة النصوص في المعلوماتية الحيوية". هذا غير صحيح؛ Perl كانت الأفضل في الماضي، لكن Python أصبحت الخيار الأكثر انتشاراً. الصحيح هو: "Perl كانت لغةً أساسية في المعلوماتية الحيوية، وما زالت العديد من الأدوات القديمة مكتوبة بها، لذلك من المفيد فهمها لقراءة السكربتات القديمة، لكن في المشاريع الجديدة يُفضل استخدام Python." اختبار بسيط: فحص تطبيقي: "هل أحتاج إلى كتابة كود جديد أم مجرد قراءة كود قديم؟".

Example: A Perl script that extracts identifiers from FASTA headers is a computational representation. Claiming that "these identifiers represent real genes with a specific function" is a biological claim not proven by the script. The script extracts text, but interpreting it biologically requires context and additional evidence. Practical test: before inferring anything biological from Perl script outputs, verify the identifiers against a reliable database.

مثال: سكربت Perl يستخرج معرفات من رؤوس FASTA هو تمثيل حسابي. الادعاء بأن "هذه المعرفات تمثل جينات حقيقية لها وظيفة معينة" هو ادعاء بيولوجي لا يثبته السكربت. السكربت يستخرج نصوصاً، لكن تفسيرها بيولوجياً يحتاج إلى سياق وأدلة إضافية. اختبار عملي: قبل أن تستنتج شيئاً بيولوجياً من مخرجات سكربت Perl، تحقق من دقة المعرفات مقابل قاعدة بيانات موثوقة.

مثال تطبيقي على سكربت Perl يعد التسلسلات:

· المدخلات: ملف FASTA (أو عدة ملفات) من سطر الأوامر.

· الافتراضات: أن الرأس يبدأ بعلامة >، وأن التسلسلات قد تكون في سطر واحد أو عدة أسطر (اعتماداً على تصميم السكربت)، وأن الملف نصي صالح.

· المخرجات: عدد التسلسلات في الملف (رقم).

· القيود: السكربت لا يتحقق من صحة التسلسلات (مثل وجود أحرف غير صالحة)، ولا يتعامل مع ملفات مضغوطة، ولا يحلل البيانات الوصفية.

Applied example of a Perl script that counts sequences:

اختبار: قبل تشغيل سكربت Perl، اقرأ الكود وفهم افتراضاته، ثم اختبره على ملف اصطناعي صغير معروف المحتوى.

· Inputs: a FASTA file (or multiple files) from the command line.

· Assumptions: that the header starts with >, that sequences may be in one or multiple lines (depending on the script's design), and that the file is valid text.

· Outputs: the number of sequences in the file (a number).

· Limitations: the script does not validate sequences (e.g., check for invalid characters), nor handle compressed files, nor parse metadata.

Test: before running a Perl script, read the code and understand its assumptions, then test it on a small synthetic file with known content.

عند استخدام سكربت Perl قديم، يجب تسجيل ما يلي:

· المصدر: من أين جاء السكربت، ومن كتبه، ومتى.

· الإصدار: إصدار Perl المستخدم، وأي إصدارات من المكتبات الخارجية إذا كانت مستخدمة.

· المعلمات: أي معاملات تستخدم عند تشغيل السكربت.

· التحويلات: ما الذي يفعله السكربت بالضبط (يمكن إضافة تعليقات داخل السكربت توضح ذلك).

When using a legacy Perl script, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السكربت وملف الإدخال لزميل بعد عام، هل يستطيع فهم ما يفعله ويعيد تشغيله بنفس النتائج دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: where the script came from, who wrote it, and when.

· Version: the Perl version used, and any external library versions if used.

· Parameters: any parameters used when running the script.

· Transformations: what the script actually does (the workflow can add comments inside the script to clarify).

Test: Practical check: "If I gave this script and input file to a colleague a year later, could they understand what it does and rerun it with the same results without my help?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على سكربت Perl في تحليل بيانات حقيقية، اختبره على ملف اصطناعي صغير معروف المحتوى. أنشئ ملف FASTA يحتوي على عدة تسلسلات، مع رؤوس بتنسيقات مختلفة (معرف بسيط، معرف مع مسافات، معرف مع أحرف خاصة). شغّل السكربت وتأكد من أنه يتعامل مع جميع الحالات بشكل صحيح. إذا اكتشفت أن السكربت يفشل في حالة معينة، فأنت تعلم أن افتراضاته لا تنطبق على بياناتك، وقد تحتاج إلى تعديله أو إلى أداة أخرى.

Before relying on a Perl script in real data analysis, test it on a small synthetic file with known content. Create a FASTA file containing several sequences, with headers in different formats (simple identifier, identifier with spaces, identifier with special characters). Run the script and verify that it handles all cases correctly. If you find that the script fails in a specific case, you know that its assumptions do not apply to the project’s data, and you may need to modify it or use another tool.

Chapter 18 — R for Statistics and Visualization

الفصل 18 — R والإحصاء والتصور

الفكرة المركزية: التحليل الإحصائي، الرسوم، التشخيص، reproducible visualization.

شرح أعمق للمفهوم

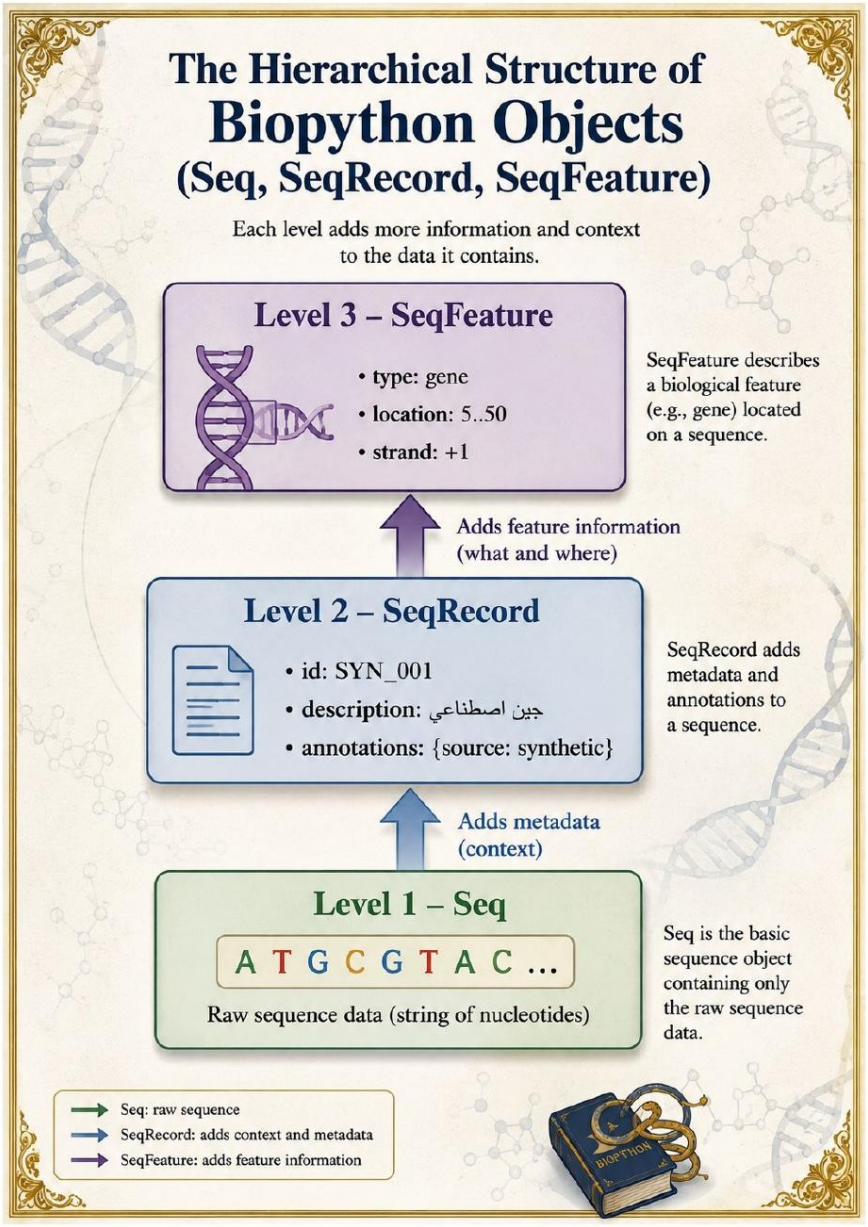
Deeper Conceptual Explanation

R لا يجعل الرسم أو الاختبار الإحصائي صحيحاً بمجرد تنفيذه. اختيار الاختبار، وبنية البيانات، والاستقلالية، وتوزيع المتغير، وتصحيح المقارنات كلها جزء من الاستدلال. التصور الجيد يكشف بنية البيانات، لكنه لا يحول النمط المرئي إلى علاقة سببية.

R does not make a statistical test or visualization correct merely because it runs. Test choice, data structure, independence, distributions, and multiple-comparison control all belong to the inference. A good visualization reveals data structure, but a visible pattern does not become causal evidence.

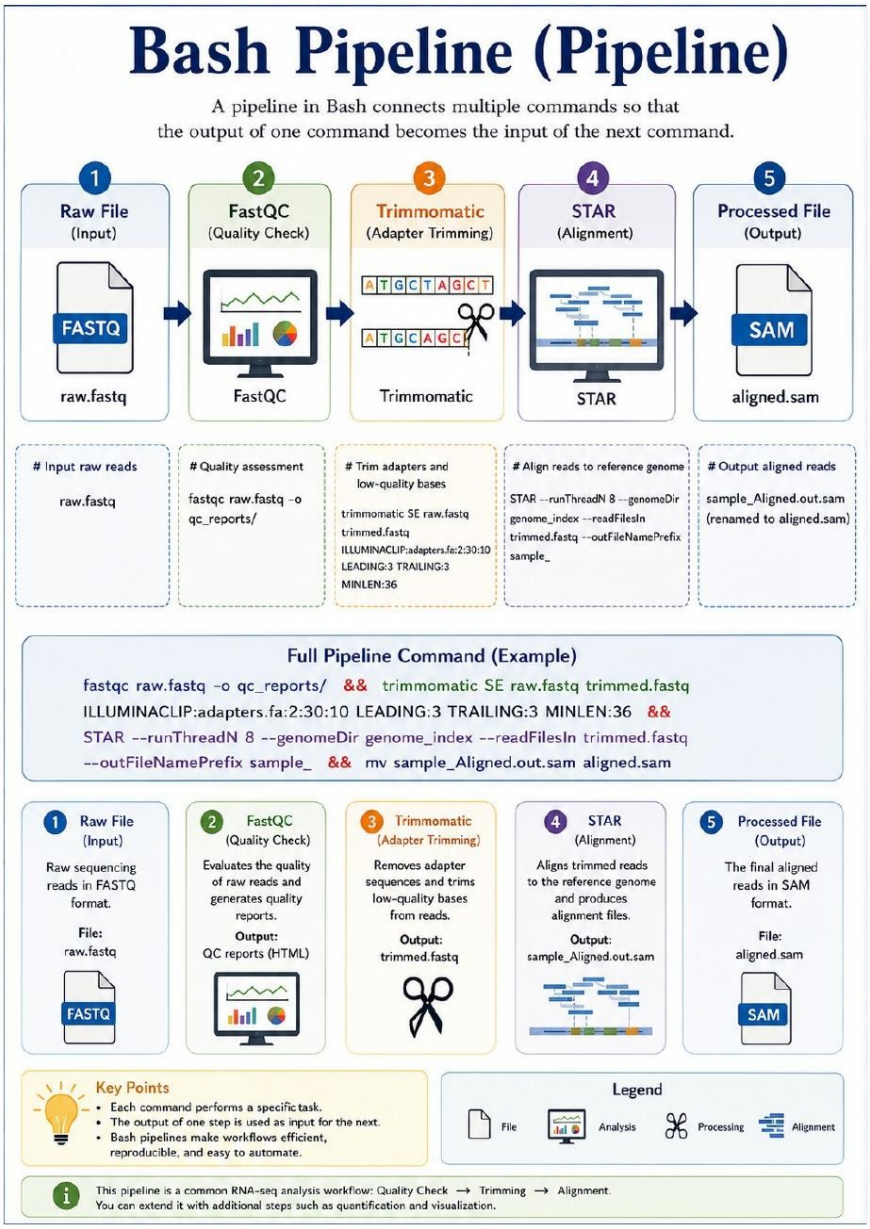
شرح أعمق للمفهوم

Deeper Conceptual Explanation


The Hierarchical Structure of
Biopython Objects
(Seq, SeqRecord, SeqFeature)
Each level adds more information and context to the data it contains.
Level 3 – SeqFeature
• type: gene
• location: 5..50
• strand: +1
SeqFeature describes a biological feature (e.g., gene) located on a sequence.
Adds feature information (what and where)
Level 2 – SeqRecord
• id: SYN_001
• description: جين اصطناعي
• annotations: {source: synthetic}
SeqRecord adds metadata and annotations to a sequence.
Adds metadata (context)
Level 1 – Seq
A T G C G T A C ...
Raw sequence data (string of nucleotides)
Seq is the basic sequence object containing only the raw sequence data.
Seq: raw sequence
SeqRecord: adds context and metadata
SeqFeature: adds feature information
BIOPYTHON

Infographic 4 / 4 إنفوجرافيك

1. Why R؟



Infographic 16 / إنفوجرافيك 16

1. لماذا R؟

R is a language specifically built for statistics and visualization. Unlike Python, which is a general-purpose language, R was designed for data analysis and visual exploration. It contains thousands of packages specialized in various areas of

biostatistics, such as gene expression analysis (RNA-seq), survival analysis, and advanced visualization using ggplot2. R is not a replacement for Python, but a complement to it. In bioinformatics, Python is often used for data processing and cleaning, while R is used for statistical analysis and visualization.

R هي لغة صُنعت خصيصاً للإحصاء والتصور. على عكس Python التي هي لغة عامة، R صُممت لتحليل البيانات واستكشافها بصرياً. تحتوي على آلاف الحزم (packages) المتخصصة في مجالات مختلفة من الإحصاء الحيوي، مثل تحليل التعبير الجيني (RNA-seq)، وتحليل البقاء (survival analysis)، والتصور المتقدم باستخدام ggplot2. R ليست بديلاً عن Python، بل هي مكملة لها. في المعلوماتية الحيوية، غالباً ما يُستخدم Python لمعالجة البيانات وتنظيفها، و R لتحليلها إحصائياً وتصورها.

```
# مثال بسيط في R: حساب المتوسط والانحراف المعياري
data <- c(2.5, 3.1, 2.8, 3.5, 2.9)

mean(data)   # المتوسط
sd(data)     # الانحراف المعياري
summary(data) # ملخص إحصائي
```

2. Visualization as a Diagnostic Tool

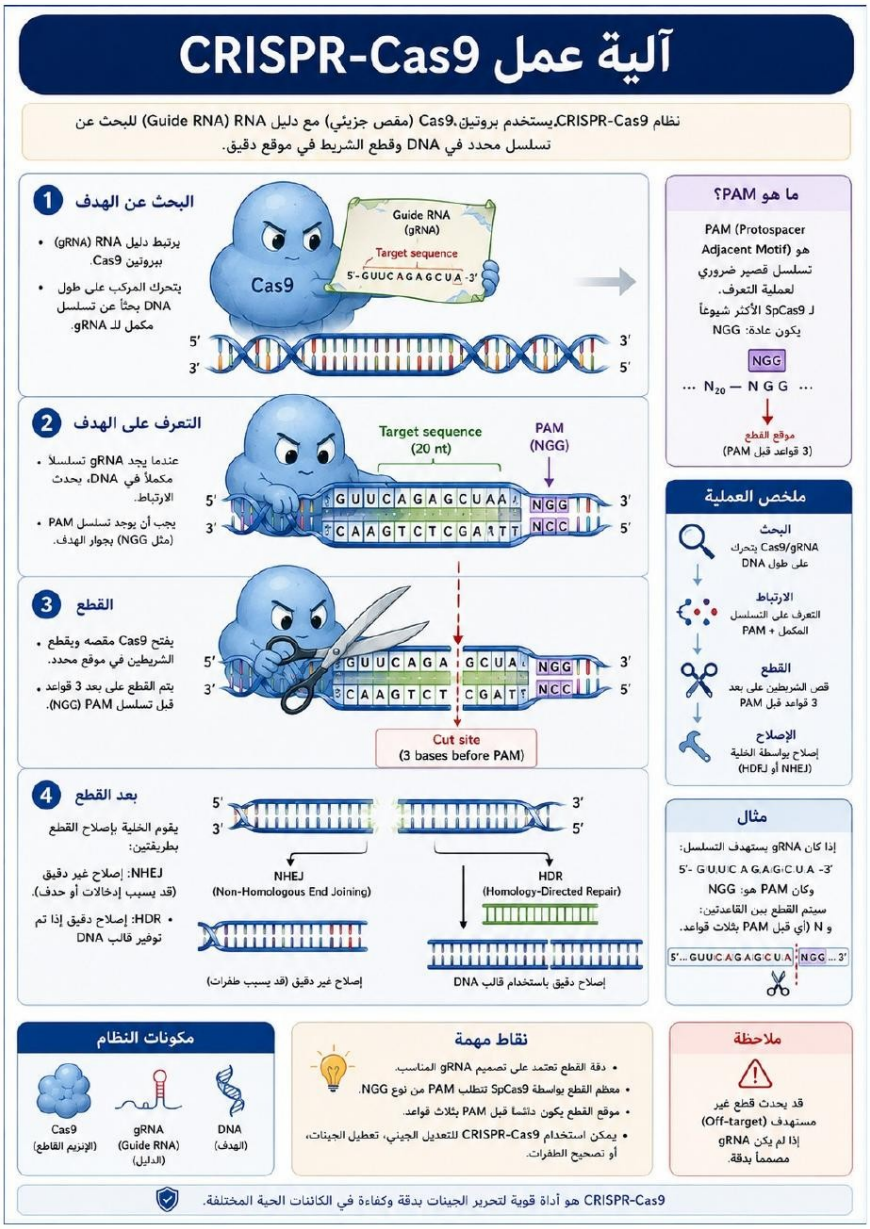
2. الرسم كأداة تشخيصية

الرقم الواحد (مثل المتوسط) قد يكون مضللاً. تخيل أن يوجد في المثال متوسطين متساويين لمجموعتين من البيانات، لكن توزيعاتهما مختلفة تماماً. الرسم البياني يكشف هذا الاختلاف. الرسم ليس مجرد خطوة نهائية لعرض النتائج؛ إنه أداة تشخيصية تساعدك على فهم بياناتك قبل أن تبدأ التحليل الإحصائي. في المعلوماتية الحيوية، الرسم يستخدم للكشف عن:

البحث الدقيق مقابل البحث التقريبي
نبحث عن نمط معين (Pattern) داخل شريط DNA
إما بتطابق تام للحروف، أو مع السماح ببعض الاختلافات.
1 البحث الدقيق (Exact Search)
النمط المطلوب (Pattern)
A T G C A G
يتوقف القطار فقط عندما يجد
تطابقاً تاما مع النمط المطلوب.
تطابق تام
(Match)
Exact Search
Exact Search
A T G C T A G C A T G C A G T T A G C G G C C G A T G C A G C
مثال:
النمط: ATGCAG
يتم العثور عليه فقط في الموقع المحدد أعلاه.
النتيجة:
تم العثور على النمط في موقع واحد فقط
حيث التطابق الكامل.
2 البحث التقريبي (Approximate Search)
النمط المطلوب (Pattern)
A T G C A G
يتوقف القطار عند المواقع التي تشبه النمط
حتى لو كان هناك اختلافات بسيطة.
تشابه عال
(High Similarity)
تشابه متوسط
(Moderate Similarity)
Approximate Search
A T G C T A G C A T G C G G T T A T G C A G G T A T G C A G C
اختلاف واحد
(1 mismatch)
اختلاف واحد
(1 mismatch)
تطابق تام
(0 mismatch)
اختلاف واحد
(1 mismatch)
مثال:
النمط: ATGCAG
تم العثور على عدة مواقع بتشابه عال أو متوسط.
النتيجة:
تم العثور على عدة مواقع تشبه النمط
بدرجة متفاوتة من التشابه.
مقارنة سريعة
النوع
شرط التطابق
عدد النتائج
الاستخدام الشائع
البحث الدقيق
(Exact Search)
تطابق تام 100٪
قليل (مواقع دقيقة)
البحث عن جينات محددة،
مواقع قص إنزيمية، علامات مميزة
البحث التقريبي
(Approximate Search)
يسمح باختلافات بسيطة
(مثل 1 أو 2 اختلاف)
متعدد (مواقع مشابهة)
البحث عن تطورات، طفرات،
جينات متشابهة
ملاحظة مهمة
البحث الدقيق أسرع ويعطي نتائج دقيقة. بينما البحث التقريبي ابطأ لكنه أكثر حساسية لاكتشاف الاختلافات.

إنفوجرافيك 17 / *Infographic 17*

· التوزيعات غير المنتظمة



Infographic 9 / 9 إنفوجرافيك

A single number (like the mean) can be misleading. Imagine having two groups of data with equal means, but their distributions are completely different. A plot reveals this difference. Visualization is not just a final step to present results; it is a

diagnostic tool that helps you understand the project's data before starting statistical analysis. In bioinformatics, visualization is used to detect:

· القيم الشاذة (outliers)

· Non-uniform distributions

· تأثيرات الدفعات (batch effects)ـ

· العلاقات غير الخطية بين المتغيراتـ

· Outliers

· Batch effects

· Non-linear relationships between variables.

```
boxplot(group1, group2, names = c("Group 1", "Group 2"),
main = "Comparison of Two Groups", ylab = "Values")
# مثال: مقارنة مجموعتين باستخدام الرسم
set.seed(123)
group1 <- rnorm(100, mean = 5, sd = 1)
group2 <- rnorm(100, mean = 5, sd = 3)
```

This simple boxplot reveals that both groups have approximately the same mean (5), but the spread is completely different (standard deviation 1 vs. 3). This difference could be critical in interpreting biological results.

هذا الرسم البسيط (boxplot) يكشف أن المجموعتين لهما نفس المتوسط تقريباً (5)، لكن التشتت مختلف تماماً (الانحراف المعياري 1 مقابل 3). هذا الاختلاف قد يكون حاسماً في تفسير النتائج البيولوجيةـ

3. Distributions and Outliers

3. التوزيعات والقيم الشاذة

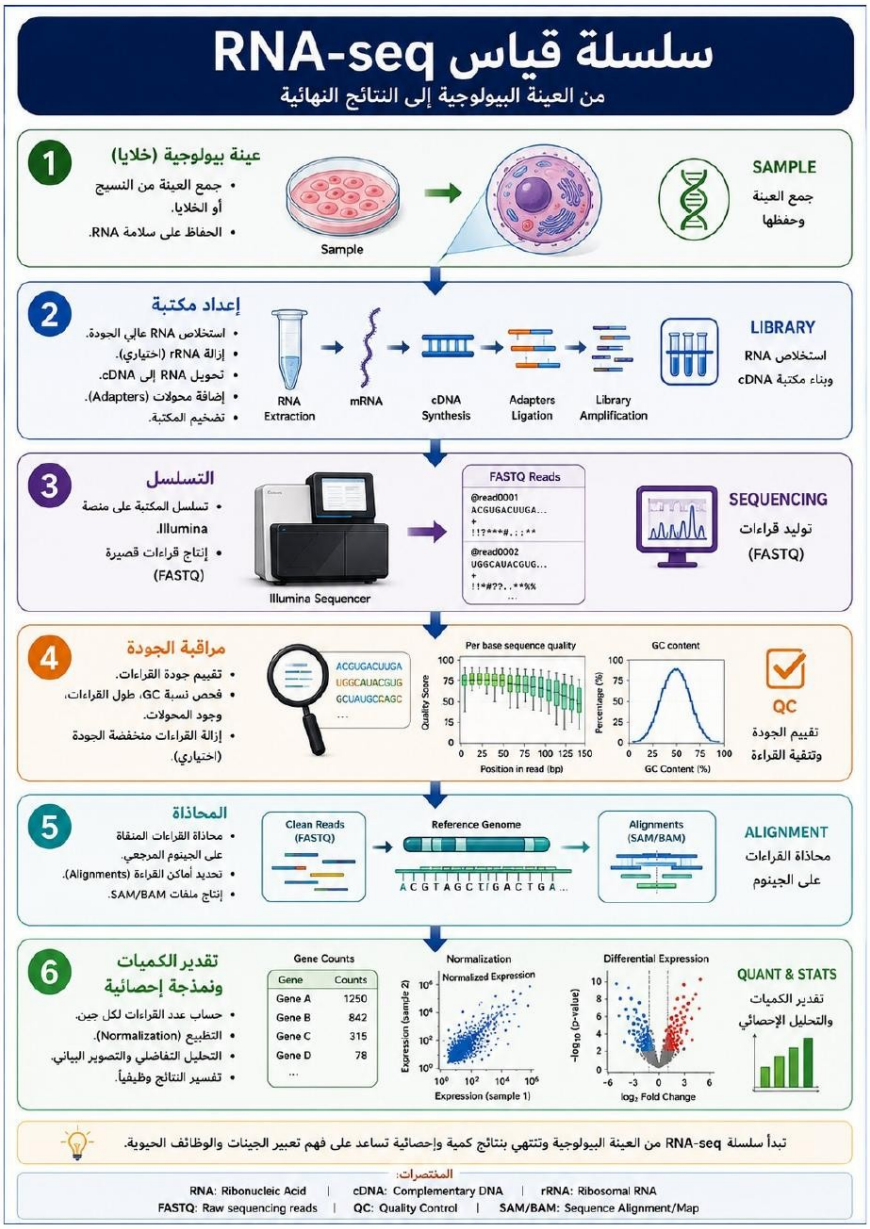
In biological data, outliers are not just errors; they may be genuine biological signals (such as exceptionally high gene expression in a diseased sample). But they may also be technical artifacts (such as a sequencing error). Visualization helps you distinguish between the two. Plots such as histograms, density plots, and Q-Q plots help you examine the distribution of the project's data and determine whether it follows a normal distribution, a fundamental assumption in many statistical tests.

في البيانات البيولوجية، القيم الشاذة ليست مجرد أخطاء؛ قد تكون إشارات بيولوجية حقيقية (مثل تعبير جيني مرتفع بشكل استثنائي في عينة مصابة). لكنها قد تكون أيضاً أخطاء تقنية (مثل خطأ في التسلسل). التصور يساعدك على

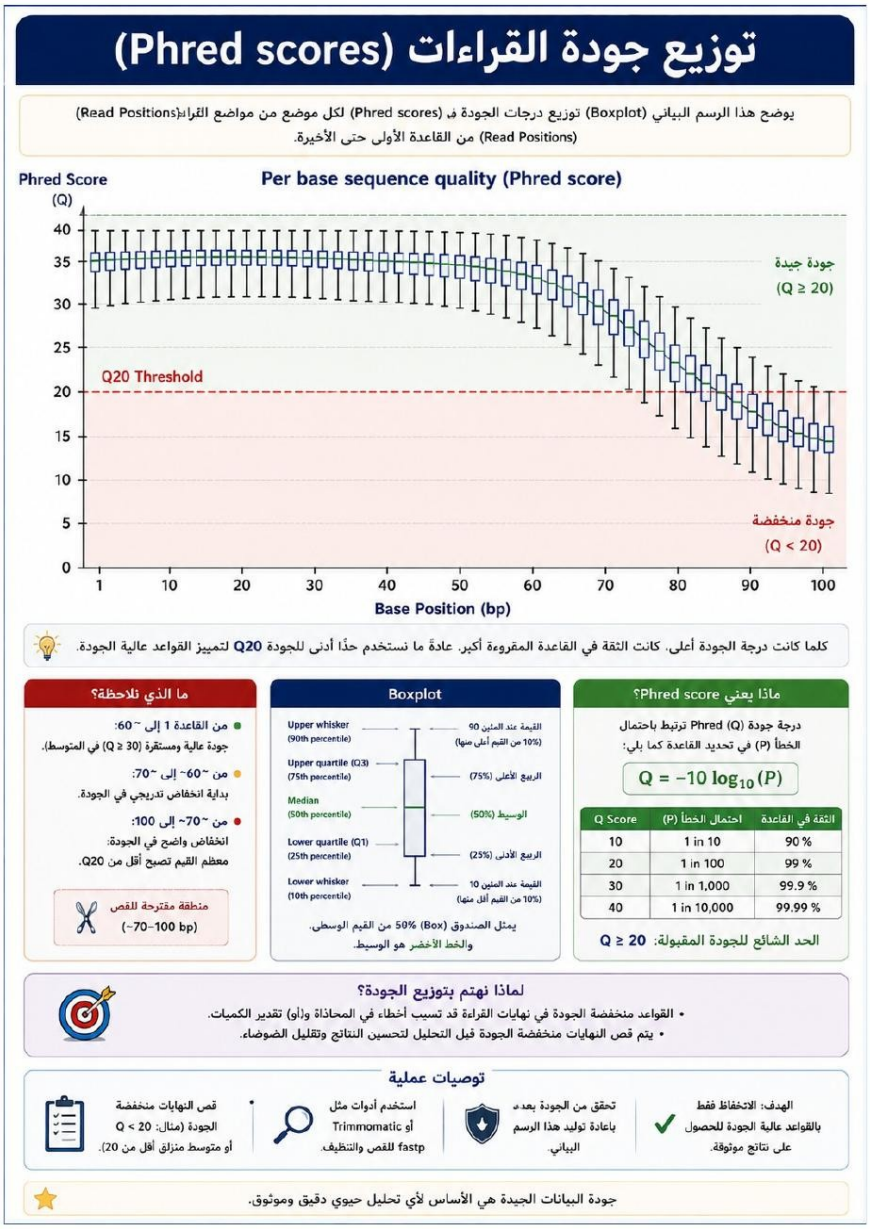
تمييز بين الحالتين. الرسوم البيانية مثل histogram و density plot و Q-Q plot تساعدك على فحص توزيع بياناتك، وتحديد ما إذا كانت تتبع توزيعاً طبيعياً أم لا، وهو افتراض أساسي في العديد من الاختبارات الإحصائية.

التمييز بين
Bioethics, Biosafety, Biosecurity
ثلاثة مفاهيم مختلفة تكنها متكاملة. هدفها المشترك: حماية الإنسان والمجتمع والبيئة مع دعم البحث العلمي المسؤول.
1
Bioethics
(أخلاقيات حيوية)
القيم والحقوق
تعنى بالتساؤلات الأخلاقية والقيمية المرتبطة بالبحث واستخدام الكائنات الحية والتقنيات الحيوية.
تركز على:
حماية كرامة الإنسان وحقوقه.
الموافقة المستنيرة والخصوصية.
العدالة والإنصاف في الوصول إلى الفوائد.
الرفق بالحيوان والاعتبارات الأخلاقية للبيئة.
المسؤولية الإجتماعية للعلماء والباحثين.
أمثلة تطبيقية:
البحوث على البشر.
تعديل الجينات.
استخدام الحيوانات في البحث.
2
Biosafety
(سلامة بيولوجية)
منع الحوادث
تعنى باتخاذ الإجراءات والضوابط للحد من التعرض غير المقصود للعوامل البيولوجية أو المواد الخطرة.
تركز على:
الحماية من المخاطر غير المقصودة.
ممارسات العمل الآمنة في المختبرات.
استخدام معدات الوقاية الشخصية (PPE).
الاحتواء المعملي وتصنيف المخاطر.
إدارة النفايات والتخلص الآمن منها.
أمثلة تطبيقية:
استخدام خزانة الأمان الحيوي (BSC)
ارتداء القفازات والمعطف المخبري.
التخلص الآمن من النفايات الحيوية.
3
Biosecurity
(أمن بيولوجي)
منع سوء الاستخدام
تعنى بالضوابط والإجراءات لمنع الوصول غير المصرح به أو الاستخدام المتعمد للعوامل البيولوجية لتحقيق ضرر.
تركز على:
منع السرقة أو الفقد أو الوصول غير المصرح به.
التحكم في نقل وتخزين المواد البيولوجية الحساسة.
التحقق من هوية الأشخاص والصلاحيات.
تقييم المخاطر المتعلقة بسوء الاستخدام المتعمد.
الامتثال للقوانين واللوائح الدولية.
أمثلة تطبيقية:
تقييد الوصول إلى المختبرات والمواد الحساسة.
التحقق من هوية العاملين.
التغليف والنقل الآمن للمواد البيولوجية.
كيف تتكامل؟
الأخلاقيات تحدد ما هو صواب.
السلامة تمنع الحوادث غير المقصودة.
الأمن يمنع الاستخدام الضار المتعمد.
التكامل بينها يضمن البحث العلمي الآمن والأخلاقي والمسؤول.
Bioethics
أخلاقيات حيوية
Biosecurity
أمن بيولوجي
Biosafety
سلامة بيولوجية
حماية الإنسان والمجتمع والبيئة ودعم البحث العلمي المسؤول
لماذا هي مهمة؟
تحمي الأفراد والباحثين والمجتمع من المخاطر.
تعزز الثقة العامة في العلم والتقنية الحيوية.
تدعم الابتكار المسؤول والمستدام.
تساهم في الأمن الصحي الوطني والعالمي.
الخلاصة
Biosecurity تعالج القيم والأخلاق، Biosafety تمنع تركز على الأمن والسلامة في المختبر، Bioethics
تمنع الوصول غير المصرح به وسوء الاستخدام. العمل المتكامل بينها أساس للتقدم العلمي الآمن والمستدام.

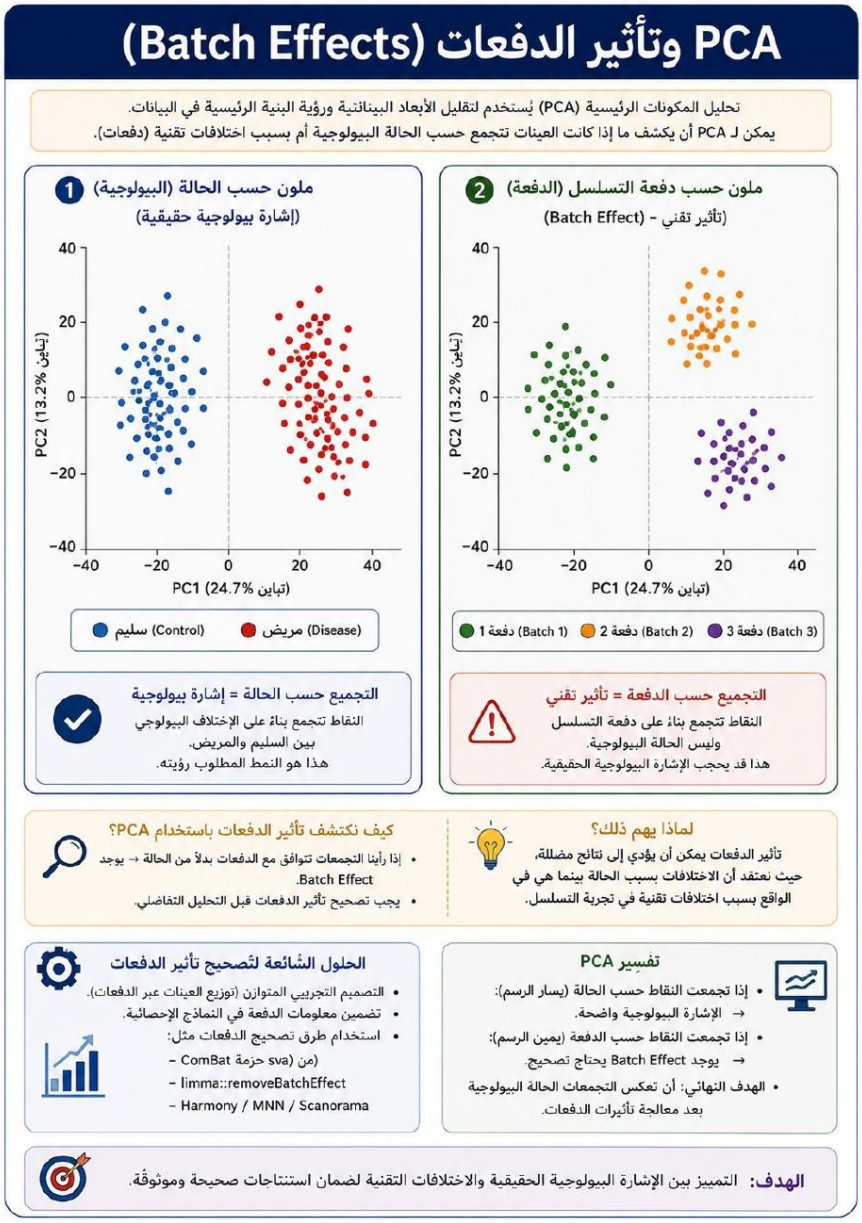
إنفوجرافيك 14 / *Infographic 14*



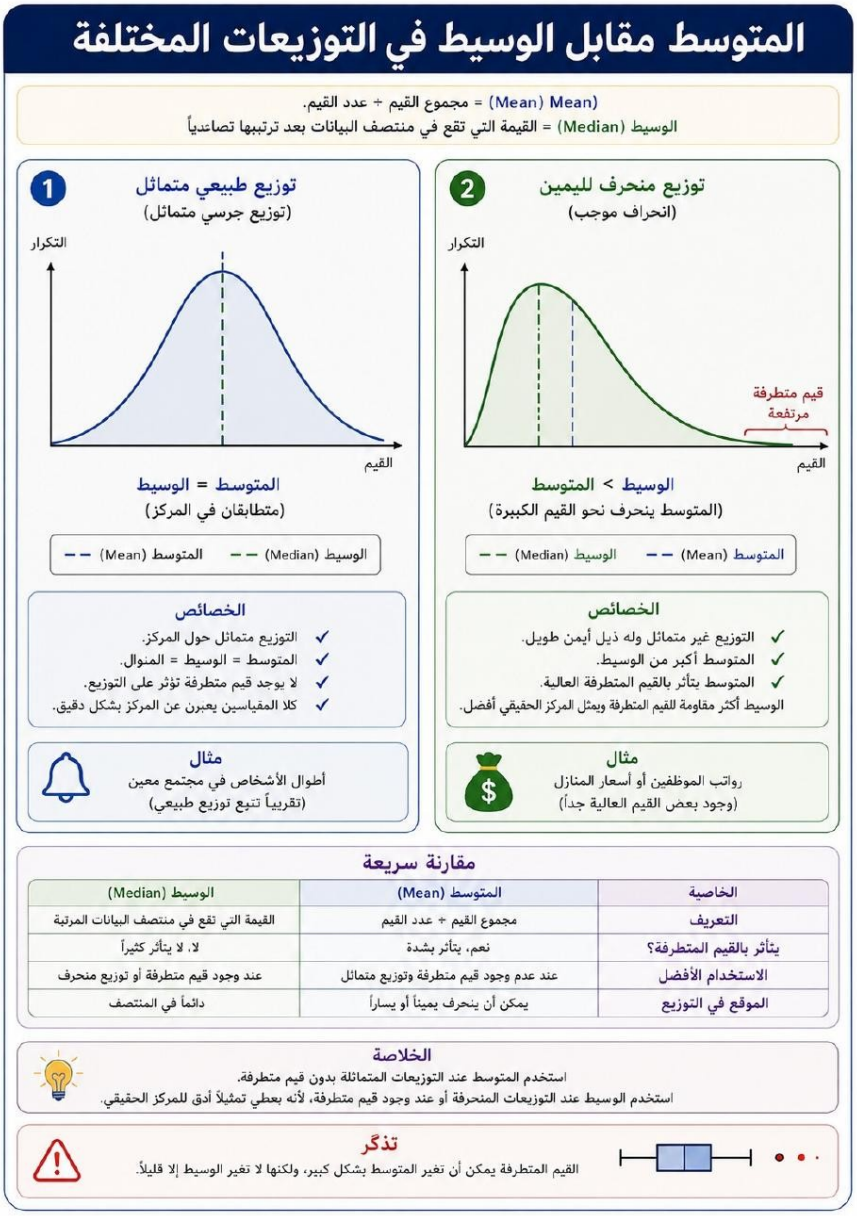
Infographic 20 / إنفوجرافيك 20



Infographic 12 / 12 إنفوجرافيك



Infographic 6 / 6 إنفوجرافيك



إنفوجرافيك 15 / *Infographic 15*

البيانات المحلية مقابل البيانات الحية
كلا النوعين من البيانات مهم وله دور مختلف. نبدأ بالبيانات المحلية (الاصطناعية) للتعلم والاختبار بأمان، ثم ننتقل إلى البيانات الحية الحقيقية عند الجاهزية، مع توثيق الإصدار ومراقبة التغييرات.
بيانات محلية
ثابتة، لا تتغير، تعمل بدون إنترنت
synthetic_data.tsv
الخصائص
بيانات اصطناعية أو مجموعة محلياً.
ثابتة، لا تتغير مع الزمن.
تعمل بدون اتصال بالإنترنت.
آمنة للتجريب والتعليم.
مثالية للاختبار، التطوير، والتعلم.
يمكن مشاركتها بسهولة.
الاستخدامات
التحقق من الأساليب
تطوير النماذج والخوارزميات
الاختبار والتجريب
التعليم والتدريب
أمثلة
ملفات CSV/TSV ثابتة.
مجموعات بيانات منشورة لا تتغير.
بيانات محاكاة (Synthetic Data).
بيانات منسوخة محلياً من إصدار معين.
ابدأ بالبيانات الاصطناعية (المحلية) للتعليم والاختبار والتأكد من صحة الطرق والنماذج.
بعد الجاهزية، انتقل إلى البيانات الحية الحقيقية مع إدارة الإصدارات ومراقبة التغييرات.
بيانات حية
تتغير باستمرار، تحتاج اتصالاً بالإنترنت، تسجيل الإصدار ضروري
الخصائص
بيانات حقيقية من مصادر متغيرة.
تتحدث باستمرار (لحظياً أو دورياً).
تحتاج اتصالاً بالإنترنت.
تتأثر بظروف الوقت والسياق.
تتطلب إدارة إصدار ومراقبة التغييرات.
أكثر دقة وتمثل الواقع.
الاستخدامات
اتخاذ قرارات مبنية على الوقت الفعلي
تشغيل النماذج في البيئة الحقيقية
لوحات المتابعة الحية
التقارير والتحليلات الحية
أمثلة
قواعد بيانات متصلة عبر الإنترنت.
بيانات من أجهزة استشعار وإنترنت الأشياء.
بيانات المستخدمين والتفاعلات الحية.
واجهات برمجة التطبيقات (APIs).
مقارنة سريعة
بيانات محلية | بيانات حية
ثابتة، لا تتغير | تتغير باستمرار
لا تحتاج | تحتاج
قد لا تمثل الواقع بدقة كاملة | تمثل الواقع بشكل أفضل
أسهل في التحكم والمشاركة | تتطلب ضوابط وصلاحيات
تعليم، إختبار، تطوير | تشغيل، تقارير، قرارات فورية
غير لازمة غالباً | ضرورية جداً
أفضل الممارسات
استخدم البيانات المحلية لفهم البيانات وبناء النماذج.
تحقق من جودة النماذج قبل تطبيقها على البيانات الحية.
عند الانتقال للبيانات الحية، وثق الإصدار والمصدر.
راقب التغيرات وابن تنبيهات عند وجود انحرافات.
حافظ على الأمان والخصوصية في جميع المراحل.
البيانات المحلية تمنحك الأمان للتعلم، والبيانات الحية تمنحك الحقيقة لاتخاذ القرار.

Infographic 28 / إنفوجرافيك 28

الرقم الواحد (مثل المتوسط) قد يكون مضللاً. تخيل أن لديك متوسطين متساويين لمجموعتين من البيانات، لكن توزيعاتهما مختلفة تماماً. الرسم البياني يكشف هذا الاختلاف. الرسم ليس مجرد خطوة نهائية لعرض النتائج؛ إنه


```
# فحص التوزيع
data <- rnorm(100, mean = 10, sd = 2)
hist(data, main = "Histogram of Data", xlab = "Values", col = "lightblue")
# إضافة قيمة شاذة متعمدة
data_with_outlier <- c(data, 50)
```

```
boxplot(data_with_outlier, main = "Boxplot with Outlier", ylab = "Values")
```

4. Reproducible Visualization

As with computational analysis, plots must be reproducible. This means that the plot should be generated from documented inputs and explicit steps, not by clicking in a graphical interface. In R, plots are created using code, and this code can be saved, shared, and rerun. This ensures that anyone else can recreate the same plot from the same data. ggplot2 is the most popular package in R for reproducible visualization, following the "grammar of graphics" principle, where you build the plot layer by layer.

4. إعادة الإنتاج في التصور (Reproducible Visualization)

كما هو الحال مع التحليل البرمجي، يجب أن تكون الرسوم البيانية قابلة لإعادة الإنتاج. هذا يعني أن الرسم يجب أن يُنشأ من بيانات موثقة وخطوات معلنة، وليس عن طريق النقر في واجهة رسومية. في R، يتم إنشاء الرسوم باستخدام كود، وهذا الكود يمكن حفظه ومشاركته وإعادة تشغيله. هذا يضمن أن أي شخص آخر يمكنه إعادة إنشاء نفس الرسم من نفس البيانات. ggplot2 هي الحزمة الأكثر شيوعاً في R للتصور القابل لإعادة الإنتاج، وهي تتبع مبدأ "طبقات الرسوم" (grammar of graphics)، حيث تبني الرسم طبقة فوق طبقة.

Purpose: perform a statistical or visualization step while separating statistical choice from mere code execution.

الغرض: تنفيذ خطوة إحصائية أو تصور للبيانات مع فصل الاختيار الإحصائي عن مجرد تشغيل الكود.

```
library(ggplot2)
# إنشاء بيانات مثال
df <- data.frame(
  sample = paste0("S", 1:20),
  expression = rnorm(20, mean = 10, sd = 2),
  condition = rep(c("control", "treatment"), each = 10)
)
# رسم قابل لإعادة الإنتاج
ggplot(df, aes(x = condition, y = expression, fill = condition)) +
  geom_boxplot() +
  geom_jitter(width = 0.2, alpha = 0.5) +
  labs(title = "Gene Expression by Condition",
       x = "Condition", y = "Expression Level") +
  theme_minimal()
```

This code produces a completely reproducible plot. Anyone with the same data and the same code will obtain the same plot.

5. Practical Application: Analyzing Gene Expression Data

هذا الكود ينتج رسمًا قابلاً لإعادة الإنتاج بالكامل. أي شخص لديه نفس البيانات ونفس الكود سيحصل على نفس الرسم.

Let us apply what we have learned to hypothetical gene expression data:

5. التطبيق العملي: تحليل بيانات تعبير جيني

لنطبق ما تعلمناه على بيانات تعبير جيني افتراضية:

```
# بيانات تعبير جيني افتراضية (10 جينات، 20 عينة)
set.seed(456)
expression_data <- matrix(rnorm(200, mean = 10, sd = 2), nrow = 10, ncol = 20)
colnames(expression_data) <- paste0("Sample", 1:20)
rownames(expression_data) <- paste0("Gene", LETTERS[1:10])
# تحويل إلى تنسيق طويل للتصور
library(reshape2)
df_long <- melt(expression_data)
colnames(df_long) <- c("Gene", "Sample", "Expression")
# رسم توزيع التعبير لكل جين
ggplot(df_long, aes(x = Gene, y = Expression, fill = Gene)) +
  geom_boxplot() +
  labs(title = "Gene Expression Distribution Across Samples",
       x = "Gene", y = "Expression Level") +
  theme_minimal() +
  theme(legend.position = "none")
# فحص القيم الشاذة لكل جين
for (gene in rownames(expression_data)) {
  vals <- expression_data[gene, ]
  q1 <- quantile(vals, 0.25)
  q3 <- quantile(vals, 0.75)
  iqr <- q3 - q1
  outliers <- vals[vals < (q1 - 1.5 * iqr) | vals > (q3 + 1.5 * iqr)]
  if (length(outliers) > 0) {
    cat("Gene", gene, "has", length(outliers), "outliers:", outliers, "\n")
  }
}
```

This analysis demonstrates how R can help explore data biologically: expression distribution, outlier detection, and generating reproducible plots.

هذا التحليل يوضح كيف يمكن لـ R أن يساعد في استكشاف البيانات بيولوجياً: توزيع التعبير، الكشف عن القيم الشاذة، وتوليد رسوم قابلة لإعادة الإنتاج.

6. Limitations of R

6. حدود R

R قوية في الإحصاء والتصور، لكن لها حدود:

· الأداء: R أبطأ من Python في معالجة كميات ضخمة من البيانات، خاصة العمليات التكرارية.

· الذاكرة: R تحمّل البيانات في الذاكرة، مما يحد من حجم البيانات التي يمكن معالجتها.

· منحنى التعلم: بناء الجملة في R قد يكون غير مألوف لمن اعتاد على Python.

R is powerful in statistics and visualization, but it has limitations:

· التكامل: بعض الأدوات تعمل فقط في Python أو فقط في R، لذا تحتاج إلى التكامل بينهما.

· Performance: R is slower than Python in handling very large datasets, especially iterative operations.

في الممارسة العملية، يستخدم الكثيرون Python لمعالجة البيانات وتنظيفها، و R لتحليلها إحصائياً وتصورها. هذا التكامل بين اللغتين هو أفضل ممارسة في المعلوماتية الحيوية.

· Memory: R loads data into memory, limiting the size of data that can be processed.

· Learning curve: R's syntax can be unfamiliar to those accustomed to Python.

· Integration: Some tools work only in Python or only in R, so the analysis requires to integrate between them.

In practice, many use Python for data processing and cleaning, and R for statistical analysis and visualization. This integration between the two languages is a best practice in bioinformatics.

Chapter Summary

خلاصة الفصل

R is a powerful environment for statistics and visualization in bioinformatics. Visualization is not just decoration, but a diagnostic and interpretive tool that reveals data distribution, outliers, batch effects, and uncertainty. Reproducible plots are generated from documented inputs and explicit steps, ensuring transparency and reliability. R complements Python: Python for processing, R for statistical analysis and visualization. The next chapter will introduce Bioconductor, the ecosystem of R packages specialized in bioinformatics.

R هي بيئة قوية للإحصاء والتصور في المعلوماتية الحيوية. الرسم البياني ليس مجرد زينة، بل أداة تشخيصية وتفسيرية تكشف توزيع البيانات، والقيم الشاذة، وتأثيرات الدفعات، وعدم اليقين. الرسوم القابلة لإعادة الإنتاج تُنشأ من بيانات موثقة وخطوات معلنة، مما يضمن الشفافية والموثوقية. R تكمل Python: Python للمعالجة، R للتحليل الإحصائي والتصور. الفصل القادم سيعرض لك Bioconductor، وهي منظومة حزم R المتخصصة في المعلوماتية الحيوية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "R for Statistics and Visualization" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "R والإحصاء والتصور" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "using R guarantees the correctness of statistical conclusions." This is incorrect; R is a tool that performs statistical operations, but the correctness of conclusions depends on data quality and the appropriateness of statistical choices. The correct statement is: "R provides powerful tools for statistical analysis and visualization, but correctly interpreting the results requires statistical understanding and biological context." A simple test: Practical check: "Did I choose the appropriate statistical test for my biological question, or did I just use the default test?".

مثال على التجاوز: القول إن "استخدام R يضمن صحة الاستنتاجات الإحصائية". هذا غير صحيح؛ R هي أداة تنفذ العمليات الإحصائية، لكن صحة الاستنتاجات تعتمد على جودة البيانات وملاءمة الاختيارات الإحصائية. الصحيح هو: "R توفر أدوات قوية لإجراء التحليل الإحصائي والتصور، لكن تفسير النتائج بشكل صحيح يتطلب فهماً إحصائياً وسياقاً بيولوجياً." اختبار بسيط: فحص تطبيقي: "هل اخترت الاختبار الإحصائي المناسب لسؤالي البيولوجي، أم أنني استخدمت الاختبار الافتراضي فقط؟"ـ

Example: A plot showing a difference in expression between two groups is a computational representation. Claiming that "this difference proves a biological effect" is a biological claim not proven by the plot alone. The plot reveals a pattern, but interpreting it biologically requires additional evidence (such as effect size, statistical significance, and replication). Practical test: before inferring a biological effect from a plot, Practical check: "Is this difference large enough to be biologically meaningful? Is it supported by appropriate statistical analysis?".

مثال: رسم بياني يظهر اختلافاً في التعبير بين مجموعتين هو تمثيل حسابي. الادعاء بأن "هذا الاختلاف يثبت وجود تأثير بيولوجي" هو ادعاء بيولوجي لا يثبته الرسم وحده. الرسم يكشف نمطاً، لكن تفسيره بيولوجياً يحتاج إلى أدلة

إضافية (مثل حجم التأثير، الأهمية الإحصائية، والتكرار). اختبار عملي: قبل أن تستنتج تأثيراً بيولوجياً من رسم، فحص تطبيقي: "هل هذا الاختلاف كبير بما يكفي ليكون ذا معنى بيولوجي؟ وهل هو مدعوم بتحليل إحصائي مناسب؟"ـ

مثال تطبيقي على تحليل إحصائي في R:

· المدخلات: بيانات عددية (مثل تعبير جيني) مع بيانات وصفية (مثل الحالة: مريض/سليم)ـ

· الافتراضات: أن البيانات تمثل عينة عشوائية، وأن التوزيع مناسب للاختبار المختار، وأن القيم المفقودة تم التعامل معها بشكل مناسبـ

· المخرجات: اختبار إحصائي (مثل t-test، ANOVA)، رسوم بيانية، مقاييس تلخيصيةـ

· القيود: R لا تتحقق من صحة الافتراضات الإحصائية تلقائياً؛ هذه مسؤولية المستخدم. الاختبار الخاطئ قد يؤدي إلى استنتاجات مضللةـ

Applied example of a statistical analysis in R:

اختبار: قبل إجراء اختبار إحصائي، تحقق من افتراضاته (مثل التوزيع الطبيعي، تجانس التباينات) باستخدام الرسوم التشخيصيةـ

· Inputs: numerical data (e.g., gene expression) with metadata (e.g., condition: patient/healthy).

· Assumptions: that the data represents a random sample, that the distribution is appropriate for the chosen test, and that missing values have been properly handled.

· Outputs: a statistical test (e.g., t-test, ANOVA), plots, summary measures.

· Limitations: R does not automatically verify statistical assumptions; this is the user's responsibility. The wrong test may lead to misleading conclusions.

Test: before performing a statistical test, verify its assumptions (e.g., normality, homogeneity of variances) using diagnostic plots.

عند استخدام R، يجب تسجيل ما يلي:

· المصدر: البيانات المستخدمة، مصدرها، وتاريخ الحصول عليهاـ

· الإصدار: إصدار R، وإصدارات الحزم المستخدمة (يمكن استخدام sessionInfo() لتسجيل كل شيء)ـ

· المعلمات: المعلمات المستخدمة في الدوال الإحصائية والرسوميةـ

· التحويلات: أي تحويلات طُبقت على البيانات قبل التحليل (مثل التطبيع، تحويل اللوغاريتم)ـ

When using R, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا الكود والبيانات لزميل بعد عام، هل يستطيع إعادة تشغيله والحصول على نفس النتائج والرسوم دون تعديلات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the data used, its source, and the date obtained.

· Version: R version, and package versions used (the workflow can use sessionInfo() to record everything).

· Parameters: parameters used in statistical and graphical functions.

· Transformations: any transformations applied to the data before analysis (e.g., normalization, log transformation).

Test: Practical check: "If I gave this code and data to a colleague a year later, could they rerun it and obtain the same results and plots without modifications?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على تحليل R في بيانات حقيقية، اختبر الكود على بيانات اصطناعية ذات خصائص معروفة. أنشئ بيانات تتبع توزيعاً طبيعياً، وتأكد من أن اختبارات R تعطي النتائج المتوقعة. ثم أضف قيماً شاذة، وتأكد من أن الرسوم التشخيصية تكتشفها. إذا نجحت هذه الاختبارات، فأنت واثق من أن تحليلك يعمل كما هو متوقع، وأي مشكلة مع بيانات حقيقية تعكس خصائص البيانات وليس خطأ في الكود.

Infographic 18 / إنفوجرافيك 18

Before relying on R analysis in real data, test the code on synthetic data with known properties. Create data that follows a normal distribution, and verify that R's tests give the expected results. Then add outliers, and verify that the diagnostic plots detect them. If these tests succeed, the workflow is confident that the project's

analysis works as expected, and any problem with real data reflects the data's properties, not an error in the code.

Part 6 — Molecular Biology for the Programmer

الباب 6 — البيولوجيا الجزيئية التي يحتاجها المبرمج

Chapter 19 — Bioconductor

الفصل

الفكرة المركزية: منظومة R الحيوية، الحزم، workflows، الإصدارات وإعادة الإنتاج.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

Bioconductor يربط البرمجة بالإحصاء والبيانات الجزيئية على نطاق واسع، ويضيف طبقة من الكائنات والحزم المتخصصة. أهمية البيئة هنا ليست في عدد الحزم، بل في توافق الحزم والإصدارات والبيانات المرجعية وسير العمل.

Bioconductor connects programming with statistics and molecular data at scale through specialized packages and data structures. Its importance is not simply the number of packages, but compatibility among packages, versions, reference data, and workflows.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is Bioconductor؟

1. ما هو Bioconductor؟

Bioconductor is an open-source project that provides software tools for analyzing genomic and biological data using the R language. Established in 2001, it now includes over 2000 specialized packages, with an active community of developers and researchers. Bioconductor is not just a package repository; it is an integrated ecosystem that follows strict principles of versioning, testing, documentation, and reproducibility. Each Bioconductor release is tied to a specific R version, and major updates are released twice a year (in April and October). This regularity in versioning ensures that the tools you use today will still work the same way if you use the same version in the future.

Bioconductor هو مشروع مفتوح المصدر يوفر أدوات برمجية لتحليل البيانات الجينومية والبيولوجية باستخدام لغة R. تأسس في عام 2001، ويضم اليوم أكثر من 2000 حزمة متخصصة، مع مجتمع نشط من المطورين والباحثين.

Bioconductor ليس مجرد مستودع حزم؛ إنه منظومة متكاملة تتبع مبادئ صارمة لإدارة الإصدارات، والاختبار، والتوثيق، وإعادة الإنتاج. كل إصدار من Bioconductor يرتبط بإصدار محدد من R، ويتم إصدار تحديثات رئيسية مرتين في السنة (في أبريل وأكتوبر). هذا الانتظام في الإصدارات يضمن أن الأدوات التي تستخدمها اليوم ستظل تعمل بنفس الطريقة إذا استخدمت نفس الإصدار في المستقبل.

```
# تثبيت Bioconductor الأساسي
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
```

```
BiocManager::install(version = "3.19")  # تثبيت إصدار محدد
```

```
# تثبيت حزمة من Bioconductor
```

```
BiocManager::install("DESeq2")
```

2. Components of the Ecosystem

2. مكونات المنظومة

The Bioconductor ecosystem consists of several integrated layers, each playing a role in the analytical workflow:

تتكون منظومة Bioconductor من عدة طبقات متكاملة، لكل منها دور في سير العمل التحليلي:

First: Analysis Packages: These are the core statistical and analysis packages. They include tools for differential expression analysis (e.g., DESeq2, limma), pathway analysis (e.g., clusterProfiler), spatial data analysis (e.g., SpatialExperiment), and sequencing data processing (e.g., Rsamtools, GenomicAlignments). Each package is designed for a specific type of data and scientific question.

أولاً: الحزم التحليلية (Analysis Packages) : هذه هي حزم الإحصاء والتحليل الأساسية. تشمل أدوات لتحليل التعبير التفاضلي (مثل DESeq2 و limma)، وتحليل المسارات (مثل clusterProfiler)، وتحليل البيانات المكانية (مثل SpatialExperiment)، ومعالجة بيانات التسلسل (مثل Rsamtools و GenomicAlignments). كل حزمة مصممة لنوع معين من البيانات وسؤال علمي محدد.

Second: Annotation Resources: These packages provide maps between genetic identifiers, pathways, genomic locations, and biological functions. For example, org.Hs.eg.db links human gene identifiers across various databases, and TxDb.Hsapiens.UCSC.hg19.knownGene provides information about genes and exons based on a specific genome assembly. Without these resources, you cannot translate numbers into biological knowledge.

ثانياً: موارد التعليقات التوضيحية (Annotation Resources) : هذه الحزم توفر خرائط بين المعرفات الجينية، والمسارات، والمواقع الجينومية، والوظائف البيولوجية. على سبيل المثال، org.Hs.eg.db تربط معرفات الجينات

البشرية بمختلف قواعد البيانات، و TxDb.Hsapiens.UCSC.hg19.knownGene توفر معلومات عن الجينات والإكسونات بناءً على تجميع معين للجينوم. بدون هذه الموارد، لا يمكن ترجمة الأرقام إلى معرفة بيولوجية.

Third: Experiment Data: Bioconductor provides packages containing real experimental data (such as airway, pasilla, cancerdata) that can be used for learning, testing packages, and comparing results. These data are an educational treasure, allowing you to try tools on real data without needing to collect it the analysis itself.

ثالثاً: بيانات التجارب (Experiment Data) : Bioconductor يوفر حزماً تحتوي على بيانات تجريبية حقيقية (مثل airway، pasilla، cancerdata) التي يمكن استخدامها للتعلم، واختبار الحزم، ومقارنة النتائج. هذه البيانات هي كنز تعليمي يسمح لك بتجربة الأدوات على بيانات حقيقية دون الحاجة إلى جمعها بنفسك.

Fourth: Workflows: Bioconductor provides fully documented workflows (such as RNAseq123, DESeq2 workflow) that explain step by step how to analyze specific data from start to finish. These workflows are the best way to learn the correct usage of packages, and are also a model for how to document the project's own analysis.

رابعاً: مسارات العمل (Workflows) : Bioconductor يوفر مسارات عمل موثقة بالكامل (مثل RNAseq123، DESeq2 workflow) التي تشرح خطوة بخطوة كيفية تحليل بيانات معينة من البداية إلى النهاية. هذه المسارات هي أفضل طريقة لتعلم الاستخدام الصحيح للحزم، وهي أيضاً نموذج لكيفية توثيق تحليلك الخاص.

3. Version Management in Bioconductor

3. إدارة الإصدارات في Bioconductor

Bioconductor follows a strict release cycle. Every six months, a new version is released (in April and October). This new version comes with package updates, bug fixes, and new features. But this also means that code that works on Bioconductor 3.18 may not work on Bioconductor 3.19 if APIs or dependencies have changed. Therefore, version management in Bioconductor is not just a luxury; it is a necessity for reproducibility. You must always record the Bioconductor version you used, the versions of major packages, and the R version itself. This recording is what allows you and others to rerun the project's analysis with the same results years later.

Bioconductor تتبع دورة إصدارات صارمة. كل ستة أشهر، يُصدر إصدار جديد (في أبريل وأكتوبر). هذا الإصدار الجديد يأتي مع تحديثات للحزم، وإصلاحات للأخطاء، وميزات جديدة. لكن هذا يعني أيضاً أن الكود الذي يعمل على

Bioconductor 3.18 قد لا يعمل على Bioconductor 3.19 إذا تغيرت واجهات برمجة التطبيقات (APIs) أو الاعتماديات. لذلك، إدارة الإصدارات في Bioconductor ليست مجرد رفاهية؛ هي ضرورة لإعادة الإنتاج. يجب أن تسجل دائماً إصدار Bioconductor الذي استخدمته، وإصدارات الحزم الرئيسية، وإصدار R نفسه. هذا التسجيل هو ما يسمح لك وللآخرين بإعادة تشغيل تحليلك بنفس النتائج بعد سنوات.

```
# تسجيل إصدار Bioconductor
BiocManager::version()

# تسجيل معلومات الجلسة
sessionInfo()
# تسجيل إصدار حزمة محددة
packageVersion("DESeq2")
```

4. Bioconductor in a Larger Context

Bioconductor is not an isolated tool. In practice, data comes from experiments, is initially processed using tools like FastQC, Trimmomatic, and Bowtie2 (in Bash), is read and cleaned using Python and Biopython, and then statistically analyzed and visualized using R and Bioconductor. Bioconductor is a layer in a larger chain. Understanding this context prevents you from believing that a Bioconductor package alone is sufficient to guarantee the correctness of the result. Data has identity, version, and context, and has undergone previous transformations. Any change in any of these layers can change the result or how it is interpreted.

4. Bioconductor في سياق أكبر

Bioconductor ليست أداة منعزلة. في الممارسة العملية، البيانات تأتي من تجارب، وتُعالج مبدئياً باستخدام أدوات مثل FastQC و Trimmomatic و Bowtie2 (في Bash)، وتُقرأ وتُنظف باستخدام Python و Biopython، ثم تُحلل إحصائياً وتُصور باستخدام R و Bioconductor. Bioconductor هي طبقة في سلسلة أكبر. فهم هذا السياق يمنعك من الاعتقاد بأن حزمة Bioconductor وحدها كافية لضمان صحة النتيجة. البيانات تحتاج إلى هوية، وإصدار، وسياق، وقد مرت بتحويلات سابقة. أي تغيير في إحدى هذه الطبقات يمكن أن يغير النتيجة أو طريقة تفسيرها.

سير العمل النموذجي:

تجربة معملية ← تسلسل (FASTQ) ← تنقية (Trimmomatic) ← محاذاة (Bowtie2) ← معالجة في Python/Biopython ← تحليل إحصائي في R/Bioconductor ← تصور وتقرير

5. Practical Application: Gene Expression Analysis Using DESeq2

Let us apply what we have learned to a real-world example using the DESeq2 package for differential expression analysis. We will use experimental data built into Bioconductor (airway):

5. التطبيق العملي: تحليل تعبير جيني باستخدام DESeq2

لنطبق ما تعلمناه على مثال واقعي باستخدام حزمة DESeq2 لتحليل التعبير التفاضلي. سنستخدم بيانات تجريبية مدمجة في Bioconductor (airway):

```
# تثبيت الحزم المطلوبة
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(c("DESeq2", "airway", "ggplot2"))

# تحميل البيانات والتعبير عنها
library(DESeq2)
library(airway)
data("airway")
# إنشاء كائن DESeq2 من بيانات التعبير الخام
dds <- DESeqDataSetFromMatrix(
  countData = assay(airway),
  colData = colData(airway),
  design = ~ cell
)

# تشغيل التحليل
dds <- DESeq(dds)

# استخراج النتائج
res <- results(dds)
res <- res[order(res$padj), ]

# عرض أول 10 جينات ذات أهمية
head(res, 10)
# رسم بياني لقيم التعبير
plotMA(res, main = "MA Plot of Differential Expression", ylim = c(-5, 5))
```

في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. في R، أسماء الكائنات وبنية الـdata frame وصيغة النموذج أهم من مجرد نجاح السطر في التنفيذ. النتيجة الإحصائية لا تُفسر قبل التأكد من أن الصفوف والأعمدة والـmetadata مرتبطة بالعينات الصحيحة.

For statistical analysis, successful execution is not sufficient. Sample names must match between the count matrix and metadata, factor levels and the design formula must be inspected, and the reference level must be explicit. Results should then be interpreted through effect size, uncertainty, and multiplicity rather than a single p-value.

في التحليل الإحصائي لا يكفي أن تنفذ الدالة دون خطأ. يجب فحص تطابق أسماء العينات بين مصفوفة العد وmetadata، وترتيب مستويات العامل، وصيغة التصميم، والمرجع المستخدم للمقارنة. ثم تُقرأ المخرجات مع حجم التأثير وعدم اليقين والتعددية، لا مع قيمة p منفردة.

This code does not turn count values into significance by itself. Each step belongs to a statistical model: object construction, design specification, parameter estimation, and extraction of the intended comparison. Coefficient names and contrasts should be inspected from the fitted model rather than guessed.

هذا الكود لا يحول أعداد العدّ إلى «دلالة» مباشرة. يجب قراءة كل خطوة بوصفها جزءًا من نموذج إحصائي: بناء الكائن، تعريف التصميم، تقدير المعلمات، ثم استخراج المقارنة المطلوبة. أسماء المعاملات والـcontrast يجب فحصها من النموذج نفسه لا تخمينها.

This simple code performs a complete differential expression analysis, from raw data to statistical results and plots. Everything is completely reproducible, provided that the versions of Bioconductor, R, and DESeq2 are recorded.

هذا الكود البسيط يقوم بتحليل كامل للتعبير التفاضلي، من البيانات الخام إلى النتائج الإحصائية والرسوم البيانية. كل شيء قابل لإعادة الإنتاج بالكامل، بشرط تسجيل إصدارات Bioconductor و R و DESeq2.

6. حدود Bioconductor

6. Limitations of Bioconductor

Bioconductor أداة قوية، لكن لها حدود يجب أن تكون على دراية بها:

· منحنى التعلم: استخدام Bioconductor يتطلب فهماً لـ R، والإحصاء الحيوي، والبيولوجيا الجزيئية. هذا قد يكون شاقاً للمبتدئين.

· إدارة الإصدارات: التزام Bioconductor الصارم بالإصدارات هو قوة، لكنه قد يكون عبئاً عندما تريد استخدام حزمة جديدة تتطلب إصداراً أحدث من Bioconductor، بينما مشروع آخر يتطلب إصداراً أقدم.

Bioconductor is a powerful tool, but it has limitations it is appropriate to be aware of:

· الذاكرة: بعض حزم Bioconductor تتعامل مع بيانات جينومية ضخمة، وقد تتطلب ذاكرة كبيرة.

· Learning curve: Using Bioconductor requires an understanding of R, biostatistics, and molecular biology. This can be challenging for beginners.

· التكامل: Bioconductor مصمم للعمل في R. إذا كان تحليلك يعتمد على Python بكثافة، فقد يكون التكامل مع Bioconductor غير سلس.

· Version management: Bioconductor's strict versioning is a strength, but it can be a burden when you want to use a new package that requires a newer Bioconductor version, while another project requires an older one.

· Memory: Some Bioconductor packages handle large genomic data and may require significant memory.

· Integration: Bioconductor is designed to work within R. If the project’s analysis heavily relies on Python, integration with Bioconductor may not be seamless.

Chapter Summary

خلاصة الفصل

Bioconductor is the integrated ecosystem for biostatistical analysis in R. It provides over 2000 specialized packages, annotation resources, experimental data, and documented workflows. Bioconductor's strict versioning makes it an ideal tool for reproducibility, but it requires awareness and precise documentation. Bioconductor is a layer in a larger analytical chain that begins with the experiment and ends with biological interpretation. Understanding this context, and documenting every step and version, is what makes the project’s analysis trustworthy and reviewable. The next chapter will return you to molecular biology, to understand the biological foundation that all these tools are processing.

Bioconductor هي المنظومة المتكاملة للتحليل الإحصائي الحيوي في R. توفر أكثر من 2000 حزمة متخصصة، وموارد تعليقات توضيحية، وبيانات تجريبية، ومسارات عمل موثقة. إدارة الإصدارات الصارمة في Bioconductor تجعله أداة مثالية لإعادة الإنتاج، لكنها تتطلب وعياً وتوثيقاً دقيقاً. Bioconductor هي طبقة في سلسلة تحليلية أكبر تبدأ من التجربة وتنتهي بالتفسير البيولوجي. فهم هذا السياق، وتوثيق كل خطوة وإصدار، هو ما يجعل تحليلك قابلاً للثقة والمراجعة. الفصل القادم سيعيدك إلى البيولوجيا الجزيئية، لتفهم الأساس البيولوجي الذي تعالجه كل هذه الأدوات.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Bioconductor" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Bioconductor" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "using Bioconductor guarantees the correctness of biological conclusions." This is incorrect; Bioconductor provides rigorous analysis tools, but the correctness of conclusions depends on data quality, the appropriateness of statistical models, and the soundness of biological

interpretation. The correct statement is: "Bioconductor provides reliable and reproducible tools for biostatistical analysis, but correctly interpreting the results requires statistical understanding and biological context." A simple test: Practical check: "Does the data I input meet the assumptions of the statistical model I used? Are the results consistent with existing biological knowledge?".

مثال على التجاوز: القول إن "استخدام Bioconductor يضمن صحة الاستنتاجات البيولوجية". هذا غير صحيح؛ Bioconductor يوفر أدوات صارمة للتحليل، لكن صحة الاستنتاجات تعتمد على جودة البيانات، وملاءمة النماذج الإحصائية، وسلامة التفسير البيولوجي. الصحيح هو: "Bioconductor يوفر أدوات موثوقة وقابلة لإعادة الإنتاج للتحليل الإحصائي الحيوي، لكن تفسير النتائج بشكل صحيح يتطلب فهماً إحصائياً وسياقاً بيولوجياً." اختبار بسيط: فحص تطبيقي: "هل البيانات التي أدخلها تلبي افتراضات النموذج الإحصائي الذي استخدمته؟ وهل النتائج متسقة مع المعرفة البيولوجية الموجودة؟"ـ

Example: The DESeq2 package produces a list of differentially expressed genes with adjusted p-values. This is a computational representation. Claiming that "these genes are the cause of the disease" is a biological claim not proven by statistical analysis alone. The analysis identifies genes with altered expression, but proving a causal role requires independent experiments. Practical test: before inferring a biological role from Bioconductor results, Practical check: "Is this change in expression large enough to be biologically meaningful? Is there other evidence supporting this role?".

مثال: حزمة DESeq2 تعطي قائمة بالجينات ذات التعبير التفاضلي مع قيم p معدلة. هذا هو تمثيل حسابي. الادعاء بأن "هذه الجينات هي المسببة للمرض" هو ادعاء بيولوجي لا يثبته التحليل الإحصائي وحده. التحليل يحدد جينات ذات تغير في التعبير، لكن إثبات الدور السببي يحتاج إلى تجارب مستقلة. اختبار عملي: قبل أن تستنتج دوراً بيولوجياً من نتائج Bioconductor، فحص تطبيقي: "هل هذا التغيير في التعبير كبير بما يكفي ليكون ذا معنى بيولوجي؟ وهل هناك أدلة أخرى تدعم هذا الدور؟"ـ

مثال تطبيقي على تحليل DESeq2:

· المدخلات: مصفوفة عدّادات (count matrix) تحتوي على أعداد القراءات لكل جين في كل عينة، وبيانات وصفية عن العينات (مثل الحالة: مريض/سليم)ـ

· الافتراضات: أن بيانات العدادات تتبع توزيعاً سلبياً ذا حدين (Negative Binomial)، وأن التطبيع قد تم بشكل مناسب، وأن التصميم التجريبي يعكس السؤال العلميـ

· المخرجات: قائمة بالجينات ذات التعبير التفاضلي، مع قيم p، وقيم p معدلة، وحجم التأثير (log2 fold change).

· القيود: التحليل لا يأخذ في الاعتبار العوامل المخفية غير المقاسة (مثل دفعات التسلسل) إلا إذا أُضيفت إلى النموذج. كما أنه لا يحدد الآلية البيولوجية.

Applied example of a DESeq2 analysis:

اختبار: قبل تشغيل DESeq2، تحقق من توزيع بيانات العدادات، وتأكد من أن العينات متجانسة، وأن التصميم مناسب. استخدم رسوم plotPCA و plotMA كأدوات تشخيصية.

· Inputs: a count matrix containing read counts for each gene in each sample, and metadata about the samples (e.g., condition: patient/healthy).

· Assumptions: that the count data follows a Negative Binomial distribution, that normalization has been properly performed, and that the experimental design reflects the scientific question.

· Outputs: a list of differentially expressed genes, with p-values, adjusted p-values, and effect sizes (log2 fold change).

· Limitations: the analysis does not account for unmeasured confounding factors (e.g., sequencing batches) unless added to the model. It also does not determine biological mechanism.

Test: before running DESeq2, check the distribution of count data, ensure samples are homogeneous, and that the design is appropriate. Use plotPCA and plotMA as diagnostic tools.

عند استخدام Bioconductor، يجب تسجيل ما يلي:

· المصدر: البيانات المستخدمة (معرفها، مصدرها، تاريخ الحصول عليها)، والإصدار المرجعي للجينوم إذا كان مستخدماً.

· الإصدار: إصدار R، وإصدار Bioconductor (BiocManager::version())، وإصدارات الحزم الرئيسية (packageVersion("DESeq2")).

· المعلمات: أي معاملات غير افتراضية استخدمتها في الدوال (مثل طريقة التطبيع، العتبات).

· التحويلات: أي خطوات قمت بها قبل التحليل (مثل فلترة الجينات منخفضة التعبير، أو تحويل البيانات).

When using Bioconductor, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا الكود وبيانات الإدخال لزميل بعد عام، مع سجل الإصدارات، هل يستطيع إعادة تشغيل التحليل والحصول على نفس النتائج دون تعديلات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the data used (its identifier, source, date obtained), and the reference genome version if used.

· Version: R version, Bioconductor version (BiocManager::version()), and main package versions (packageVersion("DESeq2")).

· Parameters: any non-default parameters used in functions (e.g., normalization method, thresholds).

· Transformations: any steps performed before analysis (e.g., filtering low-expression genes, or transforming data).

Test: Practical check: "If I gave this code and input data to a colleague a year later, along with the version log, could they rerun the analysis and obtain the same results without modifications?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على تحليل Bioconductor في بيانات حقيقية، اختبر مسار العمل على بيانات تجريبية مدمجة (مثل airway أو pasilla). شغّل التحليل الكامل من البداية إلى النهاية على هذه البيانات، وتأكد من أن النتائج (الأعداد، الرسوم، القيم) تتطابق مع التوقعات أو مع النتائج المنشورة. إذا نجح الاختبار على البيانات المدمجة، فأنت واثق من أن مسار العمل يعمل كما هو متوقع، وأي اختلاف مع بياناتك الحقيقية يعكس خصائص بياناتك (وليس خطأ في الكود).

هذه الممارسة هي المعيار الذهبي في تحليل البيانات الجينومية.

Before relying on Bioconductor analysis on real data, test the workflow on built-in experimental data (e.g., airway or pasilla). Run the complete analysis from start to finish on this data, and verify that the results (counts, plots, values) match expectations or published results. If the test succeeds on the built-in data, the workflow is confident that the workflow works as expected, and any difference with the project's real data reflects the properties of the project's data (not a code error). This practice is the gold standard in genomic data analysis.

الفكرة المركزية: المادة الجزيئية، التكامل، الاتجاهية، التعبير والترجمة.

Chapter 20 — DNA, RNA, and Protein

الفصل 20 — DNA و RNA والبروتين

شرح أعمق للمفهوم

Deeper Conceptual Explanation

السلسلة الجزيئية ليست المستوى الوحيد للمعنى. DNA وRNA والبروتين مترابطون عبر عمليات النسخ والترجمة، لكن العلاقة ليست مجرد تحويل حرفي؛ فالتنظيم، والإطار القرائي، والبدائل، والبنية، والتعديلات اللاحقة للترجمة تؤثر في المعنى البيولوجي.

A molecular sequence is not the only level of meaning. DNA, RNA, and protein are linked through transcription and translation, but the relationship is not a simple character conversion; regulation, reading frame, alternative processing, structure, and post-translational modifications affect biological meaning.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. DNA: The Repository of Genetic Information

1. DNA: مستودع المعلومات الوراثية

Deoxyribonucleic acid (DNA) is the molecule that carries genetic instructions in all living organisms, ranging from single-celled bacteria to multicellular mammals. It is found in the nucleus of eukaryotes, and in organelles such as mitochondria and chloroplasts, while in prokaryotes it is not enclosed in a membranous envelope. DNA is the cell's archive: it never leaves the nucleus, but uses messengers (mRNA) to communicate with the rest of the cell.

الحمض النووي الريبوزي منقوص الأكسجين (DNA) هو الجزيء الذي يحمل التعليمات الوراثية في جميع الكائنات الحية، من البكتيريا وحيدة الخلية إلى الثدييات متعددة الخلايا. هو موجود في نواة الخلايا حقيقية النوى، وفي العضيات مثل الميتوكوندريا والبلاستيدات الخضراء، بينما في بدائيات النوى لا يكون محاطاً بغشاء نووي. DNA هو أرشيف الخلية: لا يغادر النواة أبداً، لكنه يستخدم رسلاً (mRNA) للتواصل مع بقية الخلية.

In 1953, James Watson and Francis Crick revealed the double-helix structure of DNA. It consists of two long strands of nucleotides wound around each other, like a spiral staircase. The sugar-phosphate backbone lies on the outside, while the nitrogenous bases are stacked in the interior, like staircase steps. Each base pairs with its counterpart via hydrogen bonds.

البنية الحلزونية المزدوجة (Double-Helix Structure).

في عام 1953، كشف جيمس واتسون وفرانسيس كريك عن البنية الحلزونية المزدوجة لـ DNA. تتكون من شريطين طويلين من النيوكليوتيدات يلتفان حول بعضهما البعض، مثل سلم حلزوني. سكر الفوسفات (العمود الفقري) يقع على الخارج، بينما القواعد النيتروجينية تتكدس في الداخل، مثل درجات السلم. كل قاعدة ترتبط بقاعدة مقابل لها عبر روابط هيدروجينية.

The nitrogenous bases in DNA are four types: adenine (A), guanine (G), cytosine (C), and thymine (T). A and G are purines (two rings), while C and T are pyrimidines (one ring). The golden rule of complementarity: A pairs only with T, and G pairs only with C. This means the two strands are complementary: if one strand has the sequence AATTGGCC, the complementary strand will be TTAACCGG. This complementarity is what allows accurate DNA copying during cell division.

التكامل القاعدي (Base Complementarity).

القواعد النيتروجينية في DNA هي أربعة أنواع: الأدينين (A)، والجوانين (G)، والسيتوزين (C)، والثايمين (T). A و G هما بيورينات (حلقتان)، بينما C و T هما بيريميدينات (حلقة واحدة). قاعدة التكامل الذهبية: A لا يرتبط إلا مع T، و G لا يرتبط إلا مع C. هذا يعني أن الشريطين متكاملان: إذا كان تسلسل أحد الشريطين هو AATTGGCC، فإن الشريط المكمل سيكون TTAACCGG. هذا التكامل هو ما يسمح بنسخ DNA بدقة أثناء انقسام الخلية.

The two strands in DNA run in opposite directions: the 5′ end of one strand faces the 3′ end of the other. This antiparallel orientation is fundamental to DNA replication and other nucleic acid interactions. In Biopython, when you calculate the reverse complement (reverse_complement), the workflow is simulating this antiparallel orientation.

الاتجاهية المضادة للتوازي (Antiparallel Orientation).

2. RNA: The Messenger, Regulator, and Catalyst

الشريطان في DNA يعملان في اتجاهين متعاكسين: طرف 5' من أحد الشريطين يواجه طرف 3' من الشريط الآخر. هذا الاتجاه المضاد للتوازي هو أساسي لتضاعف DNA والتفاعلات الحمضية النووية الأخرى. في Biopython، عندما تحسب المتمم العكسي (reverse_complement)، فأنت تحاكي هذا الاتجاه المضاد للتوازي.

Ribonucleic acid (RNA) resembles DNA in many ways, but differs in three fundamental structural details:

2. RNA: الرسول والمنظم والمحفز

الحمض النووي الريبوزي (RNA) يشبه DNA في كثير من النواحي، لكنه يختلف في ثلاث تفاصيل بنيوية جوهرية:

· السكر: RNA يحتوي على ريبوز، بينما DNA يحتوي على ديوكسي ريبوز (يفقد ذرة أكسجين واحدة).

· Sugar: RNA contains ribose, while DNA contains deoxyribose (which lacks one oxygen atom).

· القاعدة: RNA يحتوي على يوراسيل (U) بدلاً من الثايمين (T).

· Base: RNA contains uracil (U) instead of thymine (T).

· البنية: RNA عادةً ما يكون أحادي الشريطة، بينما DNA مزدوج الشريطة.

· Structure: RNA is usually single-stranded, while DNA is double-stranded.

These differences make RNA more flexible and capable of performing multiple roles in the cell. The main roles of RNA include:

هذه الاختلافات تجعل RNA أكثر مرونة وقدرة على أداء أدوار متعددة في الخلية. الأدوار الرئيسية للـ RNA تشمل:

· mRNA (الرسول): يحمل الرسالة الوراثية من DNA إلى الريبوسومات، حيث تُترجم إلى بروتين. يتم نسخ mRNA من DNA في النواة، ثم يغادر إلى السيتوبلازم. إذا كان تسلسل DNA هو AATTGCGC، فإن تسلسل RNA المكمل سيكون UUAACGCG.

· tRNA (الناقل): يحمل الأحماض الأمينية إلى الريبوسومات أثناء الترجمة.

· mRNA (messenger): carries the genetic message from DNA to ribosomes, where it is translated into protein. mRNA is transcribed from DNA in the nucleus, then leaves to the cytoplasm. If the DNA sequence is AATTGCGC, the complementary RNA sequence will be UUAACGCG.

· rRNA (الريبوسومي): يشكل جزءاً رئيسياً من الريبوسومات، ويساعد في محاذاة mRNA وتحفيز تكوين الروابط الببتيدية.

· tRNA (transfer): carries amino acids to ribosomes during translation.

· miRNA (الميكرو RNA): ينظم التعبير الجيني عن طريق التداخل مع تعبير mRNAs معينة.

· rRNA (ribosomal): forms a major part of ribosomes, and helps align mRNA and catalyze peptide bond formation.

· miRNA (microRNA): regulates gene expression by interfering with the expression of certain mRNA messages.

3. Protein: The Functional Executor

3. البروتين: المنفذ الوظيفي

Proteins are polymers of amino acids. There are 20 common amino acids, each with a different R group that determines its properties. Amino acids are linked together via peptide bonds to form polypeptide chains. This chain then folds into a specific three-dimensional structure, which we call a protein.

البروتينات هي بوليمرات من الأحماض الأمينية. هناك 20 حمضاً أمينياً شائعاً، لكل منها مجموعة R مختلفة تحدد خصائصه. ترتبط الأحماض الأمينية معاً عبر روابط ببتيدية لتشكل سلاسل عديدة الببتيد. هذه السلسلة تطوى بعد ذلك إلى بنية ثلاثية الأبعاد محددة، وهي ما نسميه البروتين.

Proteins perform a wide range of cellular functions: they act as enzymes that catalyze chemical reactions, as structural components, as transport molecules, as signal receptors, and as antibodies. The diversity of protein functions is what makes life possible, and this diversity stems from the diversity of amino acid sequences, which are ultimately determined by DNA sequences.

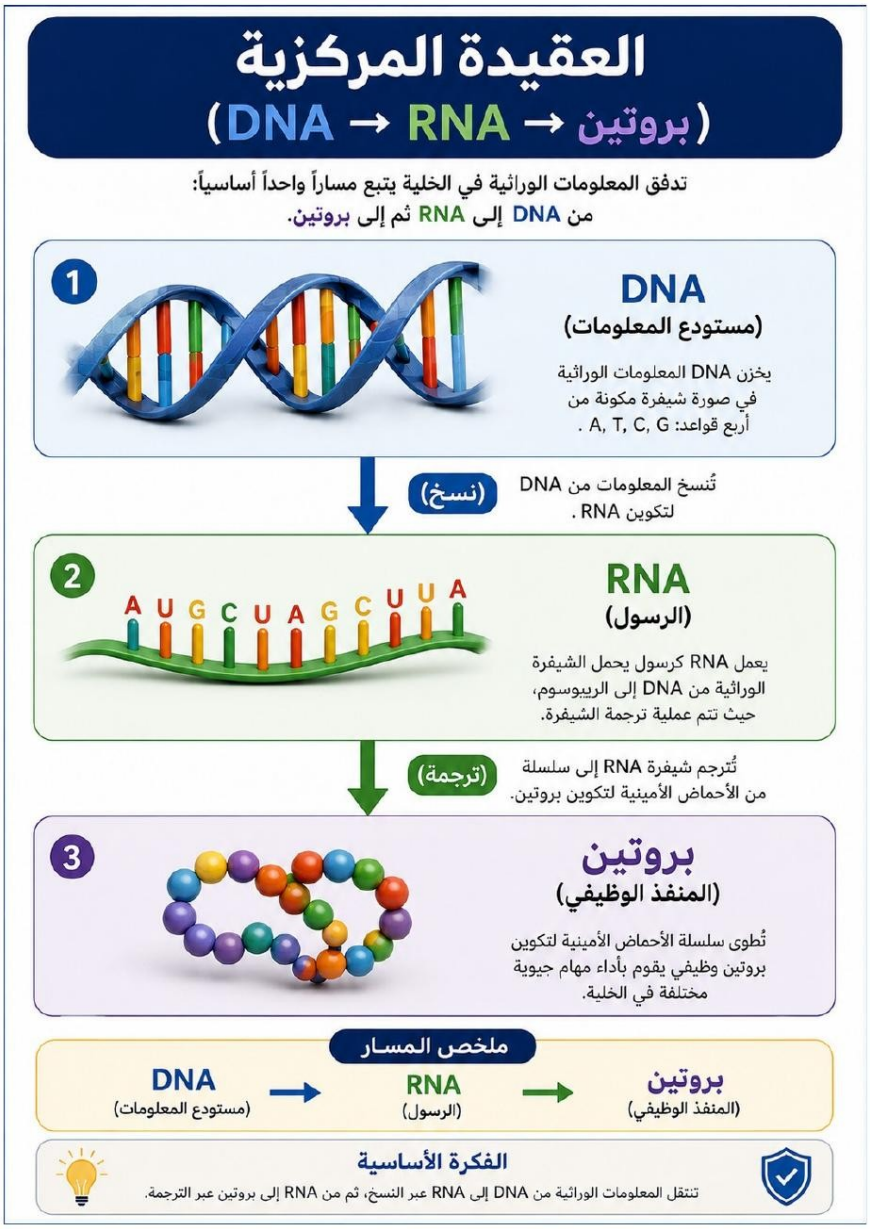
البروتينات تؤدي طيفاً واسعاً من الوظائف الخلوية: تعمل كإنزيمات تحفز التفاعلات الكيميائية، وكمكونات بنيوية، وكجزيئات نقل، وكمستقبلات للإشارات، وكأجسام مضادة. تنوع الوظائف البروتينية هو ما يجعل الحياة ممكنة، وهذا التنوع ينبع من تنوع تسلسلات الأحماض الأمينية، التي تحددها في النهاية تسلسلات DNA.

4. Transcription and Translation: The Central Dogma

4. النسخ والترجمة: العقيدة المركزية

The central dogma of molecular biology, formulated by Francis Crick in 1970, states that genetic information flows in one direction: from DNA to RNA through transcription, then from RNA to protein through translation. This process is the essence of gene expression.

العقيدة المركزية للبيولوجيا الجزيئية، التي صاغها فرانسيس كريك في 1970، تنص على أن المعلومات الوراثية تتدفق في اتجاه واحد: من DNA إلى RNA عبر النسخ، ثم من RNA إلى بروتين عبر الترجمة. هذه العملية هي جوهر التعبير الجيني.



Infographic 23 / إنفوجرافيك 23

· Transcription: A portion of DNA (a gene) is copied into mRNA in the nucleus. The transcribed segment is a complementary copy of the DNA strand, with T replaced by U.

· النسخ (Transcription): يتم نسخ جزء من DNA (جين) إلى mRNA في النواة. المقطع المُنسَخ هو نسخة مكملة لشريط DNA، مع استبدال T ب U.

· Translation: mRNA moves to the ribosome in the cytoplasm. The mRNA is read in groups of three bases called codons, each codon specifying a particular amino acid. tRNA carries the correct amino acid to the ribosome, and a polypeptide chain is formed. Translation ends at a stop codon, and the chain folds to form a functional protein.

· الترجمة (Translation): ينتقل mRNA إلى الريبوسوم في السيتوبلازم. يُقرأ mRNA في مجموعات من ثلاث قواعد تسمى الكودونات، كل كودون يحدد حمضاً أمينياً معيناً. tRNA يحمل الحمض الأميني الصحيح إلى الريبوسوم، وتتكون سلسلة عديدة الببتيد. تنتهي الترجمة عند كودون التوقف، وتطوى السلسلة لتشكل بروتيناً وظيفياً.

5. Why Does This Matter to the Programmer?

This biological understanding is not just academic curiosity. It defines the constraints of computational representation:

5. لماذا هذا مهم للمبرمج؟

هذا الفهم البيولوجي ليس مجرد فضول أكاديمي. يحدد هذا الفهم قيود التمثيل الحاسوبي:

· الاتجاه 5′→3′ يحدد لماذا reverse_complement في Biopython لا يعكس النص فحسب، بل يكمل ثم يعكس. المحاكاة يجب أن تحترم الاتجاهية.

· The 5′→3′ orientation explains why reverse_complement in Biopython does not merely reverse the text, but complements then reverses. The simulation must respect directionality.

· التكامل القاعدي (A-T، G-C) هو أساس دوال complement و translate في Biopython.

· الشفرة الوراثية (الكودون ← حمض أميني) هي جدول بحث تطبقه دالة translate().

· Base complementarity (A-T, G-C) is the foundation of the complement and translate functions in Biopython.

· البروتين ليس مجرد تسلسل؛ له بنية ثلاثية الأبعاد ووظيفة. الترجمة الحاسوبية تعطيك تسلسل الأحماض الأمينية، لكنها لا تخبرك إذا كان البروتين يطوى بشكل صحيح أو يؤدي وظيفة في الخلية. هذا هو جوهر التمييز بين العملية الحسابية والحقيقة البيولوجية.

· The genetic code (codon → amino acid) is a lookup table that the translate() function applies.

· Protein is not just a sequence; it has three-dimensional structure and function. Computational translation gives you the amino acid sequence, but it does not tell you if the protein folds correctly or performs a function in the cell. This is the essence of distinguishing between a computational operation and biological truth.

Chapter Summary

خلاصة الفصل

DNA is the repository of genetic information, with a double-helix structure, base complementarity, and antiparallel orientation. RNA is the multi-role molecule, single-stranded, that transmits, regulates, and catalyzes. Proteins are the functional executors, chains of amino acids that fold into three-dimensional structures. The central dogma (DNA → RNA → protein) describes the flow of information in the cell. Understanding these molecules and their properties is not just biological knowledge; it is the foundation that determines why we write code the way we do, and what are the limits of what computational operations can tell us about life. The next chapter will build on this foundation to understand genes and gene expression.

DNA هو مستودع المعلومات الوراثية، ببنية حلزونية مزدوجة وتكامل قاعدي واتجاه مضاد للتوازي. RNA هو الجزيء متعدد الأدوار، أحادي الشريطة، الذي ينقل المعلومات وينظمها ويحفز التفاعلات. البروتينات هي المنفذات الوظيفية، سلاسل من الأحماض الأمينية تطوى إلى بنى ثلاثية الأبعاد. العقيدة المركزية (DNA → RNA → بروتين) تصف تدفق المعلومات في الخلية. فهم هذه الجزيئات وخصائصها ليس مجرد معرفة بيولوجية؛ هو الأساس الذي يحدد لماذا نكتب الكود بالطريقة التي نكتبها، وما هي حدود ما يمكن أن تخبرنا به العمليات الحسابية عن الحياة. الفصل القادم سيبني على هذا الأساس ليفهم الجينات والتعبير الجيني.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "DNA, RNA, and Protein" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "DNA و RNA والبروتين" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "understanding DNA structure means understanding everything about the gene." This is incorrect; DNA structure tells us how information is stored, but it does not tell us about gene expression regulation or protein function. The correct statement is: "understanding DNA structure and the relationship between DNA, RNA, and protein provides the essential framework for understanding genetic information flow, but interpreting biological function requires additional context and independent experiments." A simple test: Practical

check: "Does knowing the DNA sequence suffice to determine the function of the resulting protein?".

مثال على التجاوز: القول إن "فهم بنية DNA يعني فهم كل شيء عن الجين". هذا غير صحيح؛ بنية DNA تخبرنا عن كيفية تخزين المعلومات، لكنها لا تخبرنا عن تنظيم التعبير الجيني أو الوظيفة البروتينية. الصحيح هو: "فهم بنية DNA والعلاقة بين DNA و RNA والبروتين يوفر الإطار الأساسي لفهم تدفق المعلومات الوراثية، لكن تفسير الوظيفة البيولوجية يحتاج إلى سياق إضافي وتجارب مستقلة." اختبار بسيط: فحص تطبيقي: "هل معرفة تسلسل DNA تكفي لتحديد وظيفة البروتين الناتج؟"ـ

Example: Calculating the reverse complement of a DNA sequence in Biopython is a computational representation of base complementarity and antiparallel orientation. Claiming that "this reverse complement is the second strand of DNA in the cell" is a biological claim not proven by the calculation alone. Biopython simulates the structure, but it does not confirm the existence of this strand in a real cell. Practical test: before using a computational operation as evidence of a biological structure, verify the context and independent evidence.

مثال: حساب المتمم العكسي لتسلسل DNA في Biopython هو تمثيل حسابي للتكامل القاعدي والاتجاه المضاد للتوازي. الادعاء بأن "هذا المتمم العكسي هو الشريط الثاني للـ DNA في الخلية" هو ادعاء بيولوجي لا يثبته مجرد الحساب. Biopython يحاكي البنية، لكنه لا يؤكد وجود هذا الشريط في خلية حقيقية. اختبار عملي: قبل أن تستخدم عملية حسابية كدليل على بنية بيولوجية، تحقق من السياق والأدلة المستقلةـ

مثال تطبيقي على دالة translate() في Biopython:

· القيود: translate() لا تتحقق من صحة الإطار القرائي، ولا تكتشف أخطاء التسلسل، ولا تأخذ في الاعتبار التعديلات اللاحقة للترجمة (post-translational modifications). هي أداة حسابية تطبق قواعد الشفرة الوراثية، لكنها لا تخبرنا إذا كان البروتين الناتج موجوداً أو وظيفياً في الخليةـ

· Limitations: translate() does not verify the reading frame, nor detect sequencing errors, nor account for post-translational modifications. It is a computational tool that applies the rules of the genetic code, but it does not tell us whether the resulting protein exists or is functional in the cell.

عند استخدام عمليات بيولوجية في الكود (مثل الترجمة أو حساب المتمم)، يجب تسجيل ما يلي:

· الإصدار: إصدار Biopython، وإصدار Python، وإصدار جدول الشفرة الوراثية المستخدمـ

· التحويلات: أي خطوات قمت بها قبل استخدام الدوال (مثل قص التسلسل، أو تغيير الإطار القرائي)ـ

When using biological operations in code (such as translation or complement calculation), record the following:

· Version: Biopython version, Python version, and the genetic code table version used.

· Transformations: any steps you performed before using the functions (e.g., trimming the sequence, or changing the reading frame).

قبل أن تستخدم الدوال البيولوجية في Biopython على تسلسل حقيقي، اختبرها على تسلسلات اصطناعية صغيرة معروفة النتيجة. استخدم التسلسل "ATGC" وتأكد أن reverse_complement() يعطي "GCAT". استخدم التسلسل "ATGGCC"(الذي يترجم إلى "MA") وتأكد أن translate() يعطي النتيجة المتوقعة. استخدم التسلسل "TAA" (كودون توقف) وتأكد أن translate() يتعامل معه بشكل صحيح (إما بإيقاف الترجمة أو بإرجاع علامة *). إذا نجحت هذه الاختبارات، فأنت واثق من أن الدوال تعمل كما هو متوقع، وأي نتيجة تحصل عليها من بيانات حقيقية تعكس خصائص البيانات وليس خطأ في استخدامك للمكتبة.

Before using biological functions in Biopython on a real sequence, test them on small synthetic sequences with known outcomes. Use the sequence "ATGC" and verify that reverse_complement() returns "GCAT". Use the sequence "ATGGCC" (which translates to "MA") and verify that translate() gives the expected result. Use the sequence "TAA" (a stop codon) and verify that translate() handles it correctly (either by stopping translation or returning a * symbol). If these tests succeed, the workflow is confident that the functions work as expected, and any result you get from real data reflects the properties of the data, not an error in the project's use of the library.

Chapter 21 — Genes and Gene Expression

الفصل 21 — الجين والتعبير الجيني

الفكرة المركزية: الجين، النسخ، RNA، الترجمة، تنظيم التعبير.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

الجين ليس مجرد قطعة ثابتة من DNA تُشغّل أو تُطفأ. التعبير الجيني يعتمد على السياق الخلوي والتنظيمي، وعلى مراحل من النسخ والمعالجة والترجمة. لذلك فإن قياس RNA يقدم نافذة على التعبير، لكنه لا يساوي وحده قياس البروتين أو الوظيفة النهائية.

A gene is not simply a fixed DNA segment that is on or off. Gene expression depends on cellular and regulatory context and on multiple stages from

transcription and processing to translation. RNA measurements therefore provide a window into expression, but do not by themselves equal protein abundance or final function.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is a Gene؟

1. ما هو الجين؟

A gene is a hereditary unit that carries the information needed to produce a functional product, typically a protein or a functional RNA molecule. But a gene is not just a continuous DNA string. In eukaryotes (such as humans), a gene typically consists of:

الجين هو وحدة وراثية تحمل المعلومات اللازمة لإنتاج منتج وظيفي، عادةً بروتين أو جزيء RNA وظيفي. لكن الجين

ليس مجرد سلسلة DNA متصلة. في حقيقيات النوى (مثل الإنسان)، يتكون الجين عادةً من:

· المناطق التنظيمية (Regulatory regions) : تقع قبل بداية الجين (Upstream)، وتتحكم في متى وأين يُنسخ

الجين. تشمل المحفز (promoter) ومعززات (enhancers) وعناصر تنظيمية أخرى.

· الإكسونات (Exons): الأجزاء المشفرة التي تحمل المعلومات اللازمة لإنتاج البروتين أو RNA الوظيفي.

· Regulatory regions: located upstream of the gene start, controlling when and where the gene is transcribed. These include promoters, enhancers, and other regulatory elements.

· الإنترونات (Introns): أجزاء غير مشفرة تقع بين الإكسونات، تُستأصل أثناء معالجة RNA في حقيقيات النوى.

· Exons: the coding parts that carry the information needed to produce protein or functional RNA.

· منطقة النهاية (Terminator/3' UTR) : تحدد نهاية النسخ وتؤثر على استقرار mRNA.

· Introns: non-coding parts located between exons, which are spliced out during RNA processing in eukaryotes.

· Terminator/3′ UTR: determines the end of transcription and affects mRNA stability.

This means that a gene in the computer is not a single sequence, but a set of interconnected parts. In Biopython, a gene is represented through a SeqRecord containing multiple SeqFeature objects (such as exon, intron, promoter). This hierarchical representation reflects the complex structure of a real gene.

هذا يعني أن الجين في الحاسوب ليس مجرد تسلسل واحد، بل هو مجموعة من الأجزاء المترابطة. في Biopython، يتم تمثيل الجين عبر SeqRecord الذي يحتوي على SeqFeature متعددة (مثل exon، intron، promoter). هذا التمثيل الهرمي يعكس البنية المعقدة للجين الحقيقي.

from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio.SeqFeature import SeqFeature, FeatureLocation

تمثيل جين بسيط: إكسونان وإنترون بينهما #

gene_seq = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC")
record = SeqRecord(gene_seq, id="GENE_001", description="جين اصطناعي مع إكسونات")

```
record.features.extend([exon1, intron, exon2])
# إكسون 1 (من القاعدة 0 إلى 10)
exon1 = SeqFeature(FeatureLocation(0, 10), type="exon")
# إنترون (من القاعدة 10 إلى 20)
intron = SeqFeature(FeatureLocation(10, 20), type="intron")
# إكسون 2 (من القاعدة 20 إلى 30)
exon2 = SeqFeature(FeatureLocation(20, 30), type="exon")
for feature in record.features:
  print(f"{feature.type}: {feature.location}")
```

2. Gene Expression: From Gene to Product

Gene expression is the process by which the information stored in a gene produces a functional product. It consists of two main steps, with additional layers of regulation and processing in eukaryotes:

2. التعبير الجيني: من الجين إلى المنتج

Transcription: The gene (or part of it) is copied into mRNA by the enzyme RNA polymerase. Transcription starts at the promoter and ends at the terminator region. In prokaryotes, transcription and translation occur in the same place and at roughly the same time. In eukaryotes, transcription occurs in the nucleus, then mRNA leaves to the cytoplasm.

التعبير الجيني هو العملية التي تنتج من خلالها المعلومات المخزنة في الجين منتجاً وظيفياً. تتكون من خطوتين رئيسيتين، مع طبقات إضافية من التنظيم والمعالجة في حقيقيات النوى:

النسخ (Transcription) : يتم نسخ الجين (أو جزء منه) إلى mRNA بواسطة إنزيم RNA بوليميراز. يبدأ النسخ من المحفز (promoter) وينتهي عند منطقة النهاية. في بدائيات النوى، يحدث النسخ والترجمة في نفس المكان وفي نفس الوقت تقريباً. في حقيقيات النوى، يحدث النسخ في النواة، ثم يغادر mRNA إلى السيتوبلازم.

معالجة RNA (RNA Processing) : في حقيقيات النوى، mRNA الخام (pre-mRNA) يخضع لمعالجة قبل أن يصبح جاهزاً للترجمة. تشمل المعالجة:

· إضافة قبعة 5' (5' cap) : تحمي mRNA من التحلل وتساعد في الترجمة.

RNA Processing: In eukaryotes, the raw mRNA (pre-mRNA) undergoes processing before becoming ready for translation. Processing includes:

· إضافة ذيل poly-A : يزيد من استقرار mRNA.

· 5′ cap addition: protects mRNA from degradation and aids translation.

· تضفير الإكسونات (Splicing) : تُستأصل الإنترونات، وتُربط الإكسونات معاً لتشكل mRNA ناضجاً.

· poly-A tail addition: increases mRNA stability.

· Splicing: introns are removed, and exons are joined together to form mature mRNA.

Translation: The mature mRNA is read in the ribosome, and codons are translated into a chain of amino acids, forming a protein.

الترجمة (Translation) : يُقرأ mRNA الناضج في الريبوسوم، وتُترجم الكودونات إلى سلسلة من الأحماض الأمينية، لتشكل بروتيناً.

3. Regulation of Gene Expression

The presence of a gene in the genome does not mean it is expressed at all times and in all cells. Gene expression is precisely regulated at multiple levels:

3. تنظيم التعبير الجيني

وجود جين في الجينوم لا يعني أنه يُعبَّر عنه في كل وقت وفي كل خلية. التعبير الجيني يُنظم بدقة على عدة مستويات:

· التنظيم النسخي (Transcriptional regulation) : عوامل النسخ (transcription factors) ترتبط بالمناطق التنظيمية (المحفز، المعززات) لتفعيل أو كبت النسخ. هذا هو المستوى الرئيسي للتحكم في التعبير.

· Transcriptional regulation: Transcription factors bind to regulatory regions (promoter, enhancers) to activate or repress transcription. This is the primary level of expression control.

· التنظيم بعد النسخ (Post-transcriptional regulation) : يشمل معالجة RNA، وتضفير بديل (alternative splicing) الذي يسمح لجين واحد بإنتاج بروتينات متعددة، وتنظيم استقرار mRNA بواسطة miRNAs.

· Post-transcriptional regulation: includes RNA processing, alternative splicing (which allows one gene to produce multiple proteins), and regulation of mRNA stability by miRNAs.

· التنظيم بعد الترجمة (Post-translational regulation) : البروتينات المنتجة قد تُعدّل كيميائياً (مثل الفسفرة) أو تُقطع أو تُوجه إلى مواقع محددة في الخلية.

· Post-translational regulation: produced proteins may be chemically modified (e.g., phosphorylation), cleaved, or directed to specific locations in the cell.

This regulation means that gene expression is not just an "on/off" switch. It is a spectrum of levels, changing according to cell type, developmental stage, environmental conditions, and cellular signals. From a programming perspective, this can be likened to a variable with no fixed value, but one that changes according to execution context.

4. The Central Dogma: A Useful Framework, Not an Absolute Truth

هذا التنظيم يعني أن التعبير الجيني ليس مجرد "تشغيل/إيقاف". إنه طيف من المستويات، يتغير حسب نوع الخلية، ومرحلة التطور، والظروف البيئية، والإشارات الخلوية. من منظور برمجي، يمكن تشبيه ذلك بمتغير لا قيمة ثابتة له، بل يتغير حسب سياق التنفيذ.

The central dogma of molecular biology, as formulated by Crick, states that information flow is: DNA → RNA → protein. This framework has been very useful in understanding gene expression. But it is not an absolute truth and does not cover all information pathways in the cell. There are exceptions and additional roles that make the picture more complex:

4. العقيدة المركزية: إطار مفيد، ليس حقيقة مطلقة

العقيدة المركزية للبيولوجيا الجزيئية، كما صاغها كريك، تنص على أن تدفق المعلومات هو: DNA → RNA → بروتين. هذا الإطار كان مفيداً جداً في فهم التعبير الجيني. لكنه ليس حقيقة مطلقة ولا يغطي كل مسارات المعلومات في الخلية. هناك استثناءات وأدوار إضافية تجعل الصورة أكثر تعقيداً:

· RNA كمنتج وظيفي: ليس كل RNA يُترجم إلى بروتين. بعض أنواع RNA (مثل tRNA، rRNA، miRNA) تؤدي وظائف مباشرة دون ترجمة.

· RNA as a functional product: Not all RNA is translated into protein. Some RNA types (such as miRNA, rRNA, tRNA) perform direct functions without translation.

· التدفق العكسي: بعض الفيروسات (مثل الفيروسات القهقرية) يمكن نسخ RNA إلى DNA باستخدام إنزيم النسخ العكسي (reverse transcriptase).

· Reverse flow: Some viruses (like retroviruses) can copy RNA into DNA using reverse transcriptase.

· التضفير البديل: جين واحد يمكن أن ينتج بروتينات متعددة بتضفير مختلف للإكسونات.

· Alternative splicing: One gene can produce multiple proteins through different exon splicing.

· التعديلات اللاحقة للترجمة: البروتين الناتج قد يتغير كيميائياً بعد الترجمة، مما يغير وظيفته.

· Post-translational modifications: The resulting protein may be chemically modified after translation, changing its function.

Therefore, the central dogma should be used as a historical educational framework, avoiding absolute statements that hide the diversity of molecular roles. In programming practice, this means that computational translation is just one step in a potential pathway, not the inevitable endpoint of every DNA sequence.

لذلك، يجب استخدام العقيدة المركزية كإطار تعليمي تاريخي، مع تجنب الصياغات المطلقة التي تخفي تنوع الأدوار الجزيئية. في الممارسة البرمجية، هذا يعني أن الترجمة الحاسوبية هي مجرد خطوة في مسار محتمل، وليست النهاية الحتمية لكل تسلسل DNA.

5. Practical Application: Analyzing Hypothetical Gene Expression

Let us apply what we have learned to hypothetical gene expression data. We have three genes, measuring their expression in four samples (two healthy and two diseased). We want to calculate the average expression for each gene in each condition.

5. التطبيق العملي: تحليل تعبير جيني افتراضي

لنطبق ما تعلمناه على بيانات تعبير جيني افتراضية. لدينا ثلاثة جينات، نقيس تعبيرها في أربع عينات (اثنتان سليمتان واثنتان مريضتان). نريد حساب متوسط التعبير لكل جين في كل حالة.

```
import numpy as np
# بيانات تعبير افتراضية (الجينات في الصفوف، العينات في الأعمدة)
# العينات: S1 (سليم), S2 (سليم), S3 (مريض), S4 (مريض)
expression_data = np.array([
    [10.2, 9.8, 15.3, 16.1],  # جين A
    [8.5, 8.9, 12.4, 11.8],   # جين B
```

```
    [20.1, 19.8, 22.3, 21.7]  # جين C
])

genes = ["Gene_A", "Gene_B", "Gene_C"]
conditions = ["Healthy", "Healthy", "Diseased", "Diseased"]

# حساب متوسط التعبير لكل جين في كل حالة
for i, gene in enumerate(genes):
    healthy_mean = np.mean(expression_data[i, :2])
    diseased_mean = np.mean(expression_data[i, 2:])
    fold_change = diseased_mean / healthy_mean

    print(f"{gene}:")
    print(f"  Healthy mean: {healthy_mean:.2f}")
    print(f"  Diseased mean: {diseased_mean:.2f}")
    print(f"  Fold change: {fold_change:.2f}")
    print()
```

This simple example simulates what differential expression analysis tools like DESeq2 do, but on a very small scale. The basic idea is the same: we compare expression between different conditions, and look for genes whose response changes significantly.

6. The Gene from a Programming Perspective

هذا المثال البسيط يحاكي ما تفعله أدوات تحليل التعبير التفاضلي مثل DESeq2، لكن على نطاق ضئيل جداً. الفكرة الأساسية واحدة: نقارن التعبير بين حالات مختلفة، ونبحث عن جينات تتغير استجابتها بشكل ملحوظ.

6. الجين من منظور برمجي

من منظور المبرمج، الجين يمكن تشبيهه بوظيفة (function) في الكود:

· المناطق التنظيمية هي المدخلات (parameters) التي تتحكم في متى تُستدعى الوظيفة.

· النسخ هو استدعاء الوظيفة.

· معالجة RNA هي معالجة داخلية (مثل التضفير) قد تغير الناتج.

From a programmer's perspective, a gene can be compared to a function in code:

· الترجمة هي تحويل المدخلات إلى مخرجات (بروتين).

· Regulatory regions are inputs (parameters) that control when the function is called.

· تنظيم التعبير هو التحكم في عدد مرات استدعاء الوظيفة، وشروط استدعائها.

· Transcription is calling the function.

هذا التشبيه ليس مثاليًا، لكنه يساعد في فهم أن الجين ليس مجرد بيانات ثابتة، بل هو عملية ديناميكية تعتمد على السياق.

· RNA processing is internal processing (like splicing) that may change the output.

· Translation is transforming inputs into outputs (protein).

· Regulation of expression is controlling how many times the function is called, and under what conditions.

This analogy is not perfect, but it helps in understanding that a gene is not static data, but a dynamic process that depends on context.

Chapter Summary

خلاصة الفصل

A gene is a complex hereditary unit, not just a DNA string, but including regulatory regions, exons, introns, and terminator regions. Gene expression is the process by which the information stored in a gene produces a functional product, including transcription, RNA processing, and translation. Expression regulation occurs at multiple levels, making expression dynamic and context-dependent. The central dogma (DNA → RNA → protein) is a useful framework, but it is not an absolute truth, and there are exceptions and additional roles for RNA. Understanding these complexities is what prevents the programmer from reducing biology to mere letter translation. The next chapter will move to mutations and genetic variation, and how changes in DNA sequence affect function.

الجين هو وحدة وراثية معقدة، ليست مجرد سلسلة DNA، بل تشمل مناطق تنظيمية، إكسونات، إنترونات، ومناطق نهاية. التعبير الجيني هو العملية التي تنتج من خلالها المعلومات المخزنة في الجين منتجاً وظيفياً، وتشمل النسخ، ومعالجة RNA، والترجمة. تنظيم التعبير يحدث على مستويات متعددة، مما يجعل التعبير ديناميكياً ومعتمداً على السياق. العقيدة المركزية (DNA → RNA → بروتين) هي إطار مفيد، لكنها ليست حقيقة مطلقة، وهناك استثناءات وأدوار إضافية للـ RNA. فهم هذه التعقيدات هو ما يمنع المبرمج من اختزال البيولوجيا في مجرد ترجمة حروف.

الفصل القادم سينتقل إلى الطفرات والاختلاف الوراثي، وكيف تؤثر التغييرات في تسلسل DNA على الوظيفة.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Genes and Gene Expression" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "الجين والتعبير الجيني" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "the presence of a gene in the genome means the corresponding protein is present in the cell." This is incorrect; the presence of a gene does not necessarily mean it is expressed. Expression depends on complex regulation. The correct statement is: "a gene is a hereditary unit that may or may not be expressed depending on cell type and conditions, and gene expression is a dynamic regulated process." A simple test: Practical check: "Is every gene in the human genome expressed in every cell at all times?".

مثال على التجاوز: القول إن "وجود جين في الجينوم يعني أن البروتين المقابل موجود في الخلية". هذا غير صحيح؛ وجود الجين لا يعني بالضرورة تعبيره. التعبير يعتمد على تنظيم معقد. الصحيح هو: "الجين هو وحدة وراثية قد تُعبَّر أو لا تُعبَّر حسب نوع الخلية والظروف، والتعبير الجيني هو عملية ديناميكية منظمة." اختبار بسيط: فحص تطبيقي: "هل كل جين في جينوم الإنسان يُعبَّر في كل خلية وفي كل وقت؟"ـ

Example: Calculating expression levels from RNA-seq data is a computational representation. Claiming that "high expression of this gene causes the disease" is a biological claim not proven by mere correlation. Correlation does not equal causation. Practical test: before inferring a causal relationship from expression data, Practical check: "Is this change in expression a cause of the disease or a consequence? Is there independent evidence supporting causation?".

مثال: حساب مستويات التعبير من بيانات RNA-seq هو تمثيل حسابي. الادعاء بأن "ارتفاع التعبير لهذا الجين يسبب المرض" هو ادعاء بيولوجي لا يثبته مجرد الارتباط. الارتباط لا يساوي السببية. اختبار عملي: قبل أن تستنتج علاقة سببية من بيانات تعبير، فحص تطبيقي: "هل هذا التغيير في التعبير هو سبب المرض أم نتيجة له؟ وهل هناك أدلة مستقلة تدعم السببية؟"ـ

مثال تطبيقي على تحليل تعبير افتراضي:

· المدخلات: بيانات عدادات (counts) لكل جين في كل عينة، وبيانات وصفية عن العينات (حالة، نوع خلية، وقت)ـ

· الافتراضات: أن بيانات العدادات تعكس التعبير الحقيقي، وأن التطبيع قد تم بشكل مناسب، وأن العينات ممثلة للمجتمع المراد دراستهـ

· المخرجات: متوسط التعبير لكل حالة، ونسبة التغير، وقيم p إذا أضيف اختبار إحصائيـ

· القيود: التحليل لا يأخذ في الاعتبار العوامل المخفية (مثل جودة التسلسل، اختلافات الدفعات)، ولا يحدد الآلية البيولوجيةـ

Applied example of a hypothetical expression analysis:

اختبار: قبل تحليل بيانات تعبير، تحقق من جودة البيانات (مثل توزيع العدادات، القيم الشاذة) باستخدام الرسوم التشخيصية.

· Inputs: count data for each gene in each sample, and metadata about the samples (condition, cell type, time).

· Assumptions: that the count data reflects true expression, that normalization has been properly performed, and that samples are representative of the population under study.

· Outputs: average expression per condition, fold change, and p-values if a statistical test is added.

· Limitations: the analysis does not account for confounding factors (such as sequencing quality, batch effects), nor does it determine biological mechanism.

Test: before analyzing expression data, check data quality (e.g., count distribution, outliers) using diagnostic plots.

عند تحليل بيانات تعبير جيني، يجب تسجيل ما يلي:

· المصدر: البيانات المستخدمة (معرف التجربة، الكائن الحي، النسيج، طريقة التسلسل).

· الإصدار: إصدار الأدوات المستخدمة (مثل Biopython، DESeq2)، وإصدار الجينوم المرجعي.

· المعلمات: طريقة التطبيع، والعتبات المستخدمة (مثل قيمة p، نسبة التغير).

· التحويلات: أي خطوات قمت بها لتنظيف البيانات أو تحويلها (مثل فلترة الجينات منخفضة التعبير).

When analyzing gene expression data, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة تحليل البيانات والحصول على نفس النتائج دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the data used (experiment identifier, organism, tissue, sequencing method).

· Version: versions of tools used (e.g., Biopython, DESeq2), and the reference genome version.

· Parameters: normalization method, and thresholds used (e.g., p-value, fold change).

· Transformations: any steps performed to clean or transform the data (e.g., filtering low-expression genes).

Test: Practical check: "If I gave this record to a colleague a year later, could they reanalyze the data and obtain the same results without my help?" If the answer is no, the documentation is insufficient.

قبل أن تحلل بيانات تعبير حقيقية، اختبر مسار العمل على بيانات اصطناعية صغيرة. أنشئ بيانات عدادات لجينات وهمية (5 جينات، 6 عينات، حالتان). طبّق نفس خطوات التحليل التي ستستخدمها مع البيانات الحقيقية (تطبيع، حساب متوسط، حساب نسبة التغير). تأكد من أن النتائج تتطابق مع الحسابات اليدوية. إذا نجح الاختبار، فأنت واثق من أن مسار العمل يعمل كما هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس خصائص البيانات وليس خطأ في الكود.

Before analyzing real expression data, test the workflow on small synthetic data. Create count data for hypothetical genes (5 genes, 6 samples, two conditions). Apply the same analysis steps you will use with real data (normalization, mean calculation, fold change calculation). Verify that the results match manual calculations. If the test succeeds, the workflow is confident that the workflow works as expected, and any result with real data reflects the data's properties, not an error in the code.

Chapter 22 — Mutations and Genetic Variation

الفصل 22 — الطفرات والاختلاف الوراثي

Deeper Conceptual Explanation

الفكرة المركزية: أنواع المتغيرات، أثر السياق، annotation، عدم اليقين.

شرح أعمق للمفهوم

A sequence variant does not automatically imply disease, benefit, or even a functional change. Its effect depends on location, context, type of change, affected element, population evidence, and functional evidence. Computational representation of a variant is the beginning of interpretation, not its endpoint.

الاختلاف في التسلسل لا يعني تلقائياً المرض أو المنفعة أو حتى تغير الوظيفة. أثر المتغير يعتمد على موقعه، وسياقه، ونوع التغيير، والعنصر الذي يتأثر به، والأدلة السكانية والوظيفية. التمثيل البرمجي للمتغير هو بداية للتفسير وليس نهايته.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. What Is a Genetic Variant?

1. ما هو المتغير الجيني؟

A genetic variant is any difference in a DNA sequence from a reference sequence. This difference may be small (one base) or large (a whole chromosomal rearrangement). It is important to distinguish between the terms:

المتغير الجيني هو أي اختلاف في تسلسل DNA عن تسلسل مرجعي. قد يكون هذا الاختلاف صغيراً (قاعدة واحدة) أو كبيراً (إعادة ترتيب كروموسومي كامل). من المهم التمييز بين المصطلحات:

· الطفرة (Mutation): غالباً ما يُستخدم هذا المصطلح للدلالة على تغيير جديد أو نادر، خاصة إذا كان له تأثير ضار. لكن في الممارسة العملية، المصطلحان "طفرة" و"متغير" يُستخدمان بشكل متبادل في بعض الأحيان.

· Mutation: This term is often used to indicate a new or rare change, especially if it has a harmful effect. However, in practice, "mutation" and "variant" are sometimes used interchangeably.

· الاختلاف (Polymorphism / Variant): هو تغيير موجود في أكثر من 1% من السكان. كثير من الاختلافات لا تأثير لها على الصحة أو الوظيفة.

· Polymorphism / Variant: is a change present in more than 1% of the population. Many variations have no effect on health or function.

· التبدل المفرد (SNP — Single Nucleotide Polymorphism): هو اختلاف في قاعدة واحدة بين الأفراد، وهو أكثر أنواع المتغيرات شيوعاً.

· SNP (Single Nucleotide Polymorphism): is a single-base difference between individuals, and is the most common type of variant.

```
# مثال: تمثيل متغير في Biopython
reference = "ATGCGTACGTAGCTAGCTAGCTAGCTAGC"
variant = "ATGCGTACGTAGCTAGCTAGCTAGCTAGC" # افتراض أنه مختلف في موقع واحد
# في الممارسة العملية، يتم تمثيل المتغيرات باستخدام كائنات خاصة
# أو في أدوات تحليل الجينوم Biopython مثل تلك الموجودة في
```

2. Types of Genetic Variants

Genetic variants can be classified in several ways, according to their size and nature:

2. أنواع المتغيرات الجينية

يمكن تصنيف المتغيرات الجينية بعدة طرق، حسب حجمها وطبيعتها:

أولاً: حسب الحجم (By Size) :

· متغيرات وحيدة النوكليوتيد (SNVs/Single Nucleotide Variants): تغيير في قاعدة واحدة.

· الإدخالات والحذوفات (Indels): إضافة أو حذف عدد صغير من القواعد (غالباً 1-50 قاعدة).

First: By Size:

· الاختلافات الهيكلية (Structural Variants): تغييرات كبيرة تشمل حذف أو تضاعف أو انقلاب أو إعادة ترتيب أجزاء كبيرة من الكروموسومات (أكبر من 50 قاعدة، وغالباً آلاف القواعد).

· Single Nucleotide Variants (SNVs): a change in a single base.

· Indels (Insertions/Deletions): addition or deletion of a small number of bases (usually 1-50 bases).

· Structural Variants: large changes including deletion, duplication, inversion, or rearrangement of large chromosomal segments (larger than 50 bases, often thousands of bases).

ثانياً: حسب التأثير على البروتين (By Effect on Protein) :

· متغير بمعنى تغيير (Missense): تغيير قاعدة يؤدي إلى تغيير الحمض الأميني في البروتين.

· متغير بمعنى توقف (Nonsense): تغيير قاعدة يؤدي إلى كودون توقف، مما ينتج بروتيناً مبتوراً.

Second: By Effect on Protein:

· متغير صامت (Silent): تغيير قاعدة لا يغير الحمض الأميني (بسبب تدهور الشفرة الوراثية).

· Missense variant: a base change that leads to an amino acid change in the protein.

· متغير في موقع التضفير (Splice-site): تغيير في المنطقة التي تحدد مكان تضفير الإكسونات، مما قد يؤدي إلى تضفير غير صحيح.

· Nonsense variant: a base change that leads to a stop codon, producing a truncated protein.

· Silent variant: a base change that does not change the amino acid (due to the degeneracy of the genetic code).

· Splice-site variant: a change in the region that determines where exons are spliced, which may lead to incorrect splicing.

3. Variant Detection: From Sequencing to Discovery

3. اكتشاف المتغيرات: من التسلسل إلى الاكتشاف

Variant detection is a computational process that compares a sample sequence (e.g., a patient's sequence) to a reference sequence (e.g., the human reference genome). Tools such as BWA and Bowtie2 align sequencing reads to the reference,

then tools like GATK, SAMtools, and FreeBayes identify positions where the reads differ from the reference. This process produces a VCF (Variant Call Format) file that lists the detected variants.

اكتشاف المتغيرات هو عملية حسابية تقارن تسلسل عينة (مثل تسلسل مريض) بتسلسل مرجعي (مثل الجينوم المرجعي البشري). الأدوات مثل BWA و Bowtie2 تحاذي قراءات التسلسل على المرجع، ثم أدوات مثل GATK و SAMtools و FreeBayes تحدد المواقع التي تختلف فيها القراءات عن المرجع. هذه العملية تنتج ملف VCF (Variant Call Format) الذي يسرد المتغيرات المكتشفة.

Language / نوع الكود: **VCF**

مثال على سطر في ملف VCF

```
chr1  12345  .  A  G  60  PASS  DP=50;AF=0.5  GT:AD  0/1:25,25
```

This line means: on chromosome 1, position 12345, the reference base A changed to G in the sample. Read depth (DP) is 50, allele frequency (AF) is 0.5 (half the reads carry G). The genotype (GT) is 0/1 (heterozygous).

هذا السطر يعني: في الكروموسوم 1، الموقع 12345، تغيرت القاعدة المرجعية A إلى G في العينة. عمق القراءة (DP) هو 50، والتردد الأليلي (AF) هو 0.5 (أي نصف القراءات تحمل G). النمط الجيني (GT) هو 0/1 (متغاير الزيجوت).

4. Variant ≠ Disease: The Importance of Context and Data Quality

Detecting a variant is just the beginning. The most important question is: is this variant biologically significant? Does it cause disease? The answer depends on several factors:

4. المتغير ≠ مرض: أهمية السياق وجودة البيانات

اكتشاف متغير هو مجرد بداية. السؤال الأهم هو: هل هذا المتغير ذو أهمية بيولوجية؟ هل يسبب مرضاً؟ الإجابة تعتمد على عدة عوامل:

جودة البيانات: هل المتغير مدعوم بقراءات كافية؟ هل هناك أخطاء تسلسل محتملة؟ إذا كان عمق القراءة منخفضاً، أو كانت جودة القراءات ضعيفة، فقد يكون المتغير مجرد خطأ تقني.

التردد في السكان: هل هذا المتغير موجود في السكان الأصحاء؟ إذا كان موجوداً بنسبة عالية، فمن غير المرجح أن يكون مسبباً لمرض نادر. قواعد البيانات مثل gnomAD توفر ترددات المتغيرات في مجموعات سكانية مختلفة.

Data quality: Is the variant supported by sufficient reads? Are there potential sequencing errors? If read depth is low, or read quality is poor, the variant may just be a technical artifact.

التعليق التوضيحي (Annotation) : هل يقع المتغير في منطقة مشفرة؟ هل يغير الحمض الأميني؟ هل يقع في موقع تضفير معروف؟ أدوات مثل ANNOVAR و SnpEff تضيف تعليقات للمتغيرات بناءً على مواقعها في الجينوم.

Population frequency: Is this variant present in healthy populations? If it is present at a high frequency, it is unlikely to be the cause of a rare disease. Databases like gnomAD provide variant frequencies in different population groups.

الأدلة التجريبية والسريرية: هل هناك دراسات سابقة ربطت هذا المتغير بمرض معين؟ قواعد البيانات مثل ClinVar و HGMD تجمع هذه المعلومات.

Annotation: Is the variant located in a coding region? Does it change the amino acid? Is it in a known splice site? Tools like ANNOVAR and SnpEff add annotations to variants based on their genomic locations.

Experimental and clinical evidence: Are there previous studies linking this variant to a specific disease? Databases like ClinVar and HGMD compile this information.

5. Neutral Variants

5. المتغيرات المحايدة (Neutral Variants)

معظم المتغيرات الجينية محايدة: ليس لها تأثير على الصحة أو الوظيفة. هذا لأن:

Most genetic variants are neutral: they have no effect on health or function. This is because:

· الشفرة الوراثية متدهورة (Degenerate): عدة كودونات ترمز لنفس الحمض الأميني، لذا تغيير قاعدة قد لا يغير البروتين.

· The genetic code is degenerate: several codons code for the same amino acid, so a base change may not change the protein.

· التغيير قد يحدث في منطقة غير مشفرة (Intron، منطقة بين الجينات) حيث لا يؤثر على الوظيفة.

· The change may occur in a non-coding region (intron, intergenic region) where it does not affect function.

· التغيير قد يغير حمضاً أمينياً لكنه لا يؤثر على بنية البروتين أو وظيفته.

· The change may change an amino acid but not affect protein structure or function.

Therefore, it is wrong to assume that any detected variant is disease-causing. Interpretation requires layers of evidence.

6. Practical Application: Analyzing a Simple Variant

لذلك، من الخطأ أن نفترض أن أي متغير مكتشف هو مسبب للمرض. التفسير يحتاج إلى طبقات من الأدلة.

Suppose we find a variant in the BRCA1 gene. How do we evaluate its significance?

6. التطبيق العملي: تحليل متغير بسيط

لنفترض أننا وجدنا متغيراً في جين BRCA1. كيف نقيم أهميته؟

```
# مثال توضيحي لتحليل متغير
variant_info = {
    "gene": "BRCA1",
    "position": 12345,
    "ref": "A",
    "alt": "G",
    "effect": "missense",  # تغيير حمض أميني
    "aa_change": "p.Met1Val",  # تغيير الميثيونين إلى فالين في الموقع 1
    "population_frequency": 0.00001,  # نادر جداً
    "clinvar_significance": "Pathogenic"  # مصنف على أنه ممرض في ClinVar
}

# تقييم أولي
if variant_info["population_frequency"] < 0.01:
    print("المتغير نادر في السكان.")
if variant_info["clinvar_significance"] == "Pathogenic":
    print("المتغير مصنف على أنه ممرض في ClinVar.")
if variant_info["effect"] == "missense" and variant_info["aa_change"] == "p.Met1Val":
    print("تغيير في بداية البروتين قد يكون خطيراً.")
```

This simple example shows that variant evaluation depends on integrating multiple sources of information: population frequency, annotation, and clinical databases. No single source can be relied upon alone.

7. Limits of Variant Detection and Interpretation

هذا المثال البسيط يوضح أن تقييم المتغير يعتمد على دمج مصادر متعددة للمعلومات: التردد السكاني، التعليق التوضيحي، والقواعد السريرية. لا يمكن الاعتماد على مصدر واحد.

Variant detection and interpretation have limits it is appropriate to be aware of:

7. حدود اكتشاف المتغيرات وتفسيرها

اكتشاف المتغيرات وتفسيرها له حدود يجب أن تكون على دراية بها:

· الإيجابيات الكاذبة (False Positives): قد تُكتشف متغيرات غير موجودة بسبب أخطاء في التسلسل أو المحاذاة.

· السلبيات الكاذبة (False Negatives): قد تفوت المتغيرات الحقيقية بسبب جودة تسلسل منخفضة أو تغطية غير كافية.

· False Positives: variants may be detected that do not actually exist due to sequencing or alignment errors.

· عدم اليقين في التفسير: كثير من المتغيرات تُصنف على أنها "ذات أهمية غير مؤكدة" (VUS — Variant of Uncertain Significance)، مما يعني أن الأدلة غير كافية لتصنيفها على أنها ممرضة أو حميدة.

· False Negatives: real variants may be missed due to low sequencing quality or insufficient coverage.

· تغير التصنيف بمرور الوقت: مع تراكم الأدلة، قد يتغير تصنيف المتغير من VUS إلى Pathogenic أو Benign.

· Interpretation Uncertainty: many variants are classified as VUS (Variant of Uncertain Significance), meaning evidence is insufficient to classify them as pathogenic or benign.

· Classification Changes Over Time: as evidence accumulates, a variant's classification may change from VUS to Pathogenic or Benign.

This means that variant interpretation is an ongoing process, not a final judgment. it is appropriate to treat variant detection results as hypotheses subject to revision, not as fixed facts.

هذا يعني أن تفسير المتغيرات هو عملية مستمرة، وليست حكماً نهائياً. يجب أن تتعامل مع نتائج اكتشاف المتغيرات على أنها فرضيات قابلة للتعديل، وليس حقائق ثابتة.

Chapter Summary

خلاصة الفصل

Genetic variants are differences in DNA sequence from a reference sequence. They include single nucleotide substitutions, insertions, deletions, and structural variations. Variant detection is a computational process that depends on sequencing and alignment quality. But the presence of a variant does not equal disease. Interpreting variants requires context: data quality, population frequency, annotation, and clinical evidence. Most variants are neutral, and classification may

change as evidence accumulates. Treating variants as hypotheses, and maintaining precise documentation, is the essence of responsible practice in genomics. The next chapter will move to regulation, expression, and response, to understand how cells respond to the environment by changing gene expression.

المتغيرات الجينية هي اختلافات في تسلسل DNA عن التسلسل المرجعي. تشمل التبدلات المفردة والإدخالات والحذوفات والاختلافات الهيكلية. اكتشاف المتغيرات هو عملية حسابية تعتمد على جودة التسلسل والمحاذاة. لكن وجود متغير لا يساوي مرضاً. تفسير المتغيرات يحتاج إلى سياق: جودة البيانات، التردد السكاني، التعليق التوضيحي، والأدلة السريرية. معظم المتغيرات محايدة، والتصنيف قد يتغير مع تراكم الأدلة. التعامل مع المتغيرات كفرضيات، والحفاظ على التوثيق الدقيق، هو جوهر الممارسة المسؤولة في علم الجينوم. الفصل القادم سينتقل إلى التنظيم والتعبير والاستجابة، ليفهم كيف تستجيب الخلايا للبيئة من خلال تغيير التعبير الجيني.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Mutations and Genetic Variation" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "الطفرات والاختلاف الوراثي" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "every genetic variant causes disease." This is incorrect; most variants are neutral, and some may be beneficial. The correct statement is: "genetic variants are sequence differences, some of which may affect health, but most are neutral, and interpreting their effect requires multiple lines of evidence." A simple test: Practical check: "Does every change in DNA sequence necessarily change the protein?".

مثال على التجاوز: القول إن "كل متغير جيني يسبب مرضاً". هذا غير صحيح؛ معظم المتغيرات محايدة، والبعض قد يكون مفيداً. الصحيح هو: "المتغيرات الجينية هي اختلافات في التسلسل، بعضها قد يكون له تأثير على الصحة، لكن معظمها محايد، وتفسير تأثيرها يحتاج إلى أدلة متعددة." اختبار بسيط: فحص تطبيقي: "هل كل تغيير في تسلسل DNA يغير البروتين بالضرورة؟".

Example: A VCF file listing a variant at a specific position is a computational representation. Claiming that "this variant causes cancer" is a biological claim not proven by the VCF file alone. The VCF file tells us there is a sequence difference, but it does not prove biological effect. Practical test: before inferring a biological effect from a variant, check the evidence in databases (e.g., ClinVar), data quality, and clinical context.

مثال: ملف VCF يسرد متغيراً في موقع معين هو تمثيل حسابي. الادعاء بأن "هذا المتغير يسبب السرطان" هو ادعاء بيولوجي لا يثبته ملف VCF وحده. ملف VCF يخبرنا بوجود اختلاف في التسلسل، لكنه لا يثبت التأثير البيولوجي.

اختبار عملي: قبل أن تستنتج تأثيراً بيولوجياً من متغير، تحقق من الأدلة في قواعد البيانات (مثل ClinVar)، وجودة البيانات، والسياق السريري.

مثال تطبيقي على اكتشاف متغيرات باستخدام GATK:

· المدخلات: ملف BAM (محاذاة القراءات على المرجع)، ملف مرجعي (FASTA)، قواعد بيانات التعليق التوضيحي (مثل dbsnp).

· الافتراضات: أن المحاذاة صحيحة، وأن جودة القراءات كافية، وأن الخوارزمية مضبوطة بشكل مناسب.

· المخرجات: ملف VCF يسرد المتغيرات المكتشفة مع معلومات عن الجودة والتردد.

· القيود: قد تفوت الخوارزمية متغيرات في مناطق منخفضة التغطية أو متكررة، وقد تنتج إيجابيات كاذبة بسبب أخطاء المحاذاة.

Applied example of variant detection using GATK:

اختبار: قبل اعتماد نتائج اكتشاف المتغيرات، تحقق من جودة المحاذاة، وتوزيع عمق القراءة، والمقاييس الإحصائية للخوارزمية.

· Inputs: a BAM file (reads aligned to the reference), a reference file (FASTA), annotation databases (e.g., dbSNP).

· Assumptions: that the alignment is correct, that read quality is sufficient, and that the algorithm is properly configured.

· Outputs: a VCF file listing detected variants with quality and frequency information.

· Limitations: the algorithm may miss variants in low-coverage or repetitive regions, and may produce false positives due to alignment errors.

Test: before relying on variant detection results, check alignment quality, read depth distribution, and the algorithm's statistical metrics.

عند اكتشاف المتغيرات، يجب تسجيل ما يلي:

· المصدر: البيانات الخام (ملفات FASTQ)، المرجع المستخدم (الإصدار)، وقاعدة البيانات المستخدمة للتعليق التوضيحي.

· الإصدار: إصدارات الأدوات (مثل GATK، BWA، SAMtools)، وإصدار المرجع.

· المعلمات: المعلمات المستخدمة في المحاذاة واكتشاف المتغيرات (مثل جودة العتبة، عمق القراءة الأدنى).

· التحويلات: أي خطوات لتنقية البيانات قبل اكتشاف المتغيرات (مثل إزالة القراءات المكررة، إعادة معايرة الجودة).

When detecting variants, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة اكتشاف المتغيرات والحصول على نفس النتائج دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: raw data files (FASTQ), the reference used (version), and the annotation database used.

· Version: tool versions (e.g., GATK, BWA, SAMtools), and reference version.

· Parameters: parameters used in alignment and variant detection (e.g., quality threshold, minimum read depth).

· Transformations: any data cleaning steps before variant detection (e.g., removing duplicate reads, quality recalibration).

Test: Practical check: "If I gave this record to a colleague a year later, could they re-detect the variants and obtain the same results without my help?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على أداة اكتشاف متغيرات في بيانات حقيقية، اختبر مسار العمل على بيانات اصطناعية ذات متغيرات معروفة. أنشئ تسلسلاً مرجعياً صغيراً، وأضف متغيرات معروفة (تبدلات مفردة، إدخالات، حذوفات). ولّد قراءات محاكاة، وشغّل أداة اكتشاف المتغيرات. تحقق من أن الأداة تكتشف المتغيرات المعروفة بدقة (حساسية ونوعية). إذا نجح الاختبار، فأنت واثق من أن مسار العمل يعمل كما هو متوقع، وأي اختلاف مع بيانات حقيقية يعكس خصائص البيانات وليس خطأ في الإعدادات.

Before relying on a variant detection tool on real data, test the workflow on synthetic data with known variants. Create a small reference sequence, and add known variants (SNPs, insertions, deletions). Generate simulated reads, and run the variant detection tool. Verify that the tool accurately detects the known variants (sensitivity and specificity). If the test succeeds, the workflow is confident that the workflow works as expected, and any difference with real data reflects the data's properties, not an error in the settings.

Chapter 23 — Regulation, Expression, and Response

الفصل 23 — التنظيم والتعبير والاستجابة

الفكرة المركزية: الشبكات التنظيمية، السياق الخلوي، feedback، الاستجابة.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

التعبير الجيني نتيجة لشبكة من التنظيم لا لمفتاح واحد. يمكن أن تتدخل عوامل النسخ، والإشارات الخلوية، وحالة الكروماتين، ومعالجة RNA، والاستقرار الجزيئي. عند تمثيل هذه العمليات حاسوبياً يجب أن نحدد ما الذي تم تبسيطه وما الذي لم يدخل النموذج.

Gene expression is the outcome of a regulatory network rather than a single switch. Transcription factors, cellular signals, chromatin state, RNA processing, and molecular stability can all contribute. Computational representations must state which parts were simplified and which were outside the model.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Gene Regulation: Controlling Expression

The cell does not express all its genes at all times. Gene regulation is the set of mechanisms that control when, where, and in what quantity a specific gene product is produced. This regulation occurs at multiple levels:

1. التنظيم الجيني: التحكم في التعبير

الخلية لا تُعبِّر كل جيناتها في كل وقت. التنظيم الجيني هو مجموعة الآليات التي تتحكم في متى وأين وبأي كمية يُنتج منتج جيني معين. هذا التنظيم يحدث على عدة مستويات:

التنظيم النسخي (Transcriptional regulation) : هذا هو المستوى الرئيسي للتحكم. عوامل النسخ (proteins) ترتبط بمناطق محددة على DNA (المحفزات، المعززات) لتفعيل أو كبت عملية النسخ. هذا يشبه التحكم في ما إذا كان المفتاح مفتوحاً أم مغلقاً.

التنظيم بعد النسخ (Post-transcriptional regulation) : بعد نسخ RNA، يمكن التحكم في استقراره، أو تضفيره البديل، أو نقله إلى السيتوبلازم. هذا يشبه التحكم في مدة بقاء الرسالة بعد إرسالها.

Transcriptional regulation: This is the main level of control. Transcription factors (proteins) bind to specific regions on DNA (promoters, enhancers) to activate or repress the transcription process. This is like controlling whether a switch is on or off.

التنظيم أثناء الترجمة (Translational regulation) : يمكن التحكم في معدل ترجمة mRNA إلى بروتين. هذا يشبه التحكم في سرعة قراءة الرسالة.

Post-transcriptional regulation: After RNA is transcribed, its stability, alternative splicing, or transport to the cytoplasm can be controlled. This is like controlling how long a message lasts after it is sent.

التنظيم بعد الترجمة (Post-translational regulation) : بعد إنتاج البروتين، يمكن تعديله كيميائياً (مثل الفسفرة) لتغيير نشاطه أو موقعه أو استقراره. هذا يشبه تعديل المنتج النهائي لتغيير وظيفته.

Translational regulation: The rate at which mRNA is translated into protein can be controlled. This is like controlling the reading speed of the message.

Post-translational regulation: After protein is produced, it can be chemically modified (e.g., phosphorylation) to change its activity, location, or stability. This is like modifying the final product to change its function.

2. Gene Expression: From Gene to Product

2. التعبير الجيني: من الجين إلى المنتج

Gene expression is the process by which the information stored in a gene produces a functional product. As you learned in the previous chapter, this process includes transcription, RNA processing, and translation. But in the context of regulation, expression is not a binary event (on/off), but a spectrum of levels. A gene may be expressed at a low level in one cell type, at a high level in another, and at an intermediate level under certain conditions.

التعبير الجيني هو العملية التي تنتج من خلالها المعلومات المخزنة في الجين منتجاً وظيفياً. كما تعلمت في الفصل السابق، تشمل هذه العملية النسخ، ومعالجة RNA، والترجمة. لكن في سياق التنظيم، التعبير ليس حدثاً ثنائياً (تشغيل/إيقاف)، بل هو طيف من المستويات. قد يُعبَّر الجين بمستوى منخفض في نوع معين من الخلايا، ومستوى مرتفع في نوع آخر، ومستوى متوسط في ظروف معينة.

مستويات التعبير (تمثيل بياني):

خلية طبيعية: جين A (منخفض) جين B (مرتفع) جين C (متوسط)

خلية سرطانية: جين A (مرتفع) جين B (منخفض) جين C (مرتفع)

خلية مستجيبة: جين A (متوسط) جين B (متوسط) جين C (منخفض)

This variation in expression levels is what allows the cell to respond to the environment, perform its various functions, and maintain homeostasis.

3. Sources of Expression Variation

هذا التغير في مستويات التعبير هو ما يسمح للخلية بالاستجابة للبيئة، وأداء وظائفها المختلفة، والحفاظ على التوازن الداخلي.

When you compare expression levels between two samples, you may find a difference. But this difference may arise from multiple sources:

3. مصادر الاختلاف في التعبير

عندما تقارن مستويات التعبير بين عينتين، قد تجد اختلافاً. لكن هذا الاختلاف قد ينشأ من مصادر متعددة:

الاختلاف البيولوجي (Biological variation) : هذا هو الاختلاف الحقيقي الذي يهمنا. ينشأ من اختلاف في نوع الخلية، أو الحالة المرضية، أو الاستجابة لمحفز. مثلاً، خلية سرطانية تعبّر عن جينات معينة بشكل مختلف عن خلية سليمة. هذا هو ما نريد اكتشافه.

Biological variation: This is the real difference that interests us. Arises from differences in cell type, disease state, or response to a stimulus. For example, a cancer cell expresses certain genes differently from a healthy cell. This is what we want to discover.

الاختلاف التقني (Technical variation) : ينشأ من خطوات التجربة نفسها: جودة التسلسل، كمية RNA المستخرجة، اختلافات بين دفعات التسلسل (batch effects)، اختلافات بين المختبرات. هذا هو "الضوضاء" التي نحتاج إلى تصفيتها.

Technical variation: Arises from the experimental steps themselves: sequencing quality, amount of RNA extracted, batch effects, differences between laboratories. This is the "noise" we need to filter out.

اختلافات المعالجة (Processing variation) : تنشأ من خطوات المعالجة الحاسوبية بعد التسلسل: اختيار طريقة التطبيع، خوارزمية المحاذاة، معايير التصفية. اختيارات مختلفة قد تؤدي إلى نتائج مختلفة.

Processing variation: Arises from computational processing steps after sequencing: choice of normalization method, alignment algorithm, filtering criteria. Different choices may lead to different results.

الاختلاف الكلي = اختلاف بيولوجي + اختلاف تقني + اختلاف معالجة

The goal of good experimental design and statistical analysis is to minimize and account for technical variation, and isolate the true biological variation.

4. The Importance of Experimental Design and Replication

الهدف من التصميم التجريبي الجيد والتحليل الإحصائي هو تقليل الاختلاف التقني ومعالجته، وعزل الاختلاف البيولوجي الحقيقي.

Statistical analysis cannot correct problems in experimental design. If samples are not representative, or if there are not enough replicates, or if metadata is missing, even the best statistical algorithm will not be able to distinguish signal from noise.

4. أهمية التصميم التجريبي والتكرار

لا يمكن للتحليل الإحصائي أن يصحح مشاكل في التصميم التجريبي. إذا كانت العينات غير ممثلة، أو إذا لم تكن هناك مكررات كافية، أو إذا كانت البيانات الوصفية مفقودة، فحتى أفضل خوارزمية إحصائية لن يمكن تمييز الإشارة عن الضوضاء.

المكررات البيولوجية (Biological replicates) : هي عينات مستقلة من مصادر بيولوجية مختلفة (مثل مريضين مختلفين، أو خلايا من مزرعتين مختلفتين). المكررات البيولوجية تسمح بتقدير التباين الطبيعي بين الأفراد، وهي ضرورية للاستدلال الإحصائي.

Biological replicates: are independent samples from different biological sources (e.g., two different patients, or cells from two different cultures). Biological replicates allow estimation of natural variation between individuals, and are essential for statistical inference.

المكررات التقنية (Technical replicates) : هي قياسات متكررة لنفس العينة البيولوجية (مثل تشغيل نفس العينة على جهاز التسلسل مرتين). المكررات التقنية تساعد في تقدير التباين التقني، لكنها لا تعوض عن نقص المكررات البيولوجية.

Technical replicates: are repeated measurements of the same biological sample (e.g., running the same sample on the sequencer twice). Technical replicates help estimate technical variation, but they do not compensate for a lack of biological replicates.

البيانات الوصفية (Metadata) : هي معلومات عن كل عينة (نوع الخلية، الحالة، الظروف، وقت القياس). بدون بيانات وصفية دقيقة، لا يمكن تفسير الاختلافات في التعبير.

Metadata: is information about each sample (cell type, condition, time of measurement). Without accurate metadata, expression differences cannot be interpreted.

5. Practical Application: Designing a Hypothetical Gene Expression Experiment

Suppose we want to study the effect of a new drug on gene expression in cancer cells. How would we design the experiment?

5. التطبيق العملي: تصميم تجربة تعبير جيني افتراضية

لنفترض أننا نريد دراسة تأثير دواء جديد على التعبير الجيني في خلايا سرطانية. كيف نصمم التجربة؟

Design:

1. Treatment group (exposed to drug) and control group (not exposed to drug).
2. 3 biological replicates per group (cells from 3 different cultures).
3. RNA extraction from each sample (6 samples total).
4. RNA-seq for each sample.
5. Collect metadata: cell type, drug concentration, exposure time, sequencing date, sequencing batch.

التصميم:

1. مجموعة معالجة (تتعرض للدواء) ومجموعة ضابطة (لا تتعرض للدواء).
2. 3 مكررات بيولوجية لكل مجموعة (خلايا من 3 مزارع مختلفة).
3. استخلاص RNA من كل عينة (6 عينات إجمالاً).
4. تسلسل RNA (RNA-seq) لكل عينة.
5. جمع بيانات وصفية: نوع الخلية، تركيز الدواء، وقت التعرض، تاريخ التسلسل، دفعة التسلسل.

With this design, we can use differential expression analysis (such as DESeq2) to compare the two groups, while accounting for technical differences (such as sequencing batch).

مع هذا التصميم، يمكننا استخدام تحليل التعبير التفاضلي (مثل DESeq2) لمقارنة المجموعتين، مع التحكم في الاختلافات التقنية (مثل دفعة التسلسل).

6. Distinguishing Biological from Technical Effects

One of the most important skills in expression data analysis is distinguishing between what is biological and what is technical. This requires:

6. التمييز بين التأثير البيولوجي والتأثير التقني

من أهم مهارات تحليل بيانات التعبير هي التمييز بين ما هو بيولوجي وما هو تقني. هذا يتطلب:

فحص الجودة (Quality control) : قبل أي تحليل إحصائي، فحص جودة البيانات. الرسوم البيانية مثل PCA و heatmaps يمكن أن تكشف عن تجمعات غير متوقعة (مثل تجمع العينات حسب دفعة التسلسل بدلاً من حسب الحالة). هذا يشير إلى وجود تأثير تقني.

Quality control: Before any statistical analysis, check data quality. Plots such as PCA and heatmaps can reveal unexpected clustering (e.g., samples clustering by sequencing batch rather than by condition). This indicates a technical effect.

التطبيع (Normalization) : تطبيع البيانات لتصحيح الاختلافات التقنية (مثل عمق التسلسل). اختيار طريقة التطبيع المناسبة يعتمد على نوع البيانات وتوزيعها.

Normalization: Normalize data to correct for technical differences (such as sequencing depth). Choosing the appropriate normalization method depends on the data type and distribution.

النمذجة الإحصائية (Statistical modeling) : استخدام نماذج تأخذ في الاعتبار المصادر المختلفة للتباين. في DESeq2، على سبيل المثال، يتم تضمين العوامل التقنية في النموذج الإحصائي.

Statistical modeling: Use models that account for different sources of variation. In DESeq2, for example, technical factors can be included in the statistical model.

7. Response: Dynamic Change in Expression

Response is the change in gene expression in response to an external or internal stimulus. The response may be rapid (minutes) or slow (hours or days). It may be transient or permanent. Understanding response requires measuring expression at multiple time points, not just at a single point.

7. الاستجابة: التغيير الديناميكي في التعبير

الاستجابة هي التغيير في التعبير الجيني استجابة لمحفز خارجي أو داخلي. قد تكون الاستجابة سريعة (دقائق) أو بطيئة (ساعات أو أيام). قد تكون عابرة أو دائمة. فهم الاستجابة يتطلب قياس التعبير في عدة نقاط زمنية، وليس فقط في نقطة واحدة.

Language / نوع الكود: **Text / Time-course Example**

استجابة نموذجية لمحفز:

```
t=0: تعبير أساسي (بدون محفز)
```

t=15 دقيقة: ارتفاع سريع لجينات الاستجابة المبكرة

t=60 دقيقة: ارتفاع لجينات الاستجابة المتأخرة

t=24 ساعة: عودة بعض الجينات إلى المستوى الأساسي

Understanding this dynamics requires designing time-course experiments, and performing time-series analysis.

فهم هذه الديناميكية يتطلب تصميم تجارب زمنية، وتحليل البيانات الزمنية (time-series analysis).

Chapter Summary

خلاصة الفصل

Gene regulation is the set of mechanisms that control when, where, and in what quantity a gene is expressed. Gene expression is not a binary event, but a spectrum of levels. Expression differences between samples may arise from biological, technical, or processing sources. Good experimental design, with sufficient biological replicates and accurate metadata, is the foundation for distinguishing signal from noise. Distinguishing biological from technical effects requires quality control, normalization, and statistical modeling. Response is the dynamic change in expression in response to stimuli, and requires time-course measurements. The next chapter will move to biological algorithms, starting with sequence search.

التنظيم الجيني هو مجموعة الآليات التي تتحكم في متى وأين وبأي كمية يُعبَّر الجين. التعبير الجيني ليس حدثاً ثنائياً، بل طيف من المستويات. الاختلافات في التعبير بين العينات قد تنشأ من مصادر بيولوجية أو تقنية أو معالجة. التصميم التجريبي الجيد، مع مكررات بيولوجية كافية وبيانات وصفية دقيقة، هو الأساس لتمييز الإشارة عن الضوضاء. التمييز بين التأثير البيولوجي والتأثير التقني يتطلب فحص الجودة، التطبيع، والنمذجة الإحصائية. الاستجابة هي التغيير الديناميكي في التعبير استجابة لمحفزات، وتتطلب قياسات زمنية. الفصل القادم سينتقل إلى الخوارزميات الحيوية، بدءاً بالبحث في التسلسلات.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Regulation, Expression, and Response" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "التنظيم والتعبير والاستجابة" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "every difference in expression between two samples is a true biological difference." This is incorrect; the difference may be technical (sequencing quality, batches, processing differences). The correct statement is: "expression differences may arise from multiple sources (biological, technical, processing), and distinguishing between them requires good experimental design and careful statistical analysis." A simple test: Practical check:

"Is this expression difference consistent across independent replicates, or does it appear in only one sample?".

مثال على التجاوز: القول إن "كل اختلاف في التعبير بين عينتين هو اختلاف بيولوجي حقيقي". هذا غير صحيح؛ الاختلاف قد يكون تقنياً (جودة التسلسل، دفعات، اختلافات في المعالجة). الصحيح هو: "الاختلاف في التعبير قد ينشأ من مصادر متعددة (بيولوجية، تقنية، معالجة)، والتمييز بينها يتطلب تصميماً تجريبياً جيداً وتحليلاً إحصائياً دقيقاً." اختبار بسيط: فحص تطبيقي: "هل هذا الاختلاف في التعبير ثابت عبر مكررات مستقلة، أم أنه يظهر في عينة واحدة فقط؟"ـ

Example: DESeq2 analysis results give a list of differentially expressed genes with p-values. This is a computational representation. Claiming that "these genes cause the disease" is a biological claim not proven by statistical analysis alone. The analysis identifies genes with changed expression, but proving a causal role requires independent experiments. Practical test: before inferring a biological role, Practical check: "Is there independent evidence (such as gene knockout experiments, or other studies) supporting this role?".

مثال: نتيجة تحليل DESeq2 تعطي قائمة بالجينات ذات التعبير التفاضلي مع قيم p. هذا هو تمثيل حسابي. الادعاء بأن "هذه الجينات هي المسببة للمرض" هو ادعاء بيولوجي لا يثبته التحليل الإحصائي وحده. التحليل يحدد جينات ذات تغير في التعبير، لكن إثبات الدور السببي يحتاج إلى تجارب مستقلة. اختبار عملي: قبل أن تستنتج دوراً بيولوجياً، فحص تطبيقي: "هل هناك أدلة مستقلة (مثل تجارب تعطيل الجين، أو دراسات أخرى) تدعم هذا الدور؟"ـ

مثال تطبيقي على تحليل DESeq2:

· المدخلات: مصفوفة عدادات (counts) لكل جين في كل عينة، وبيانات وصفية عن العينات (الحالة، دفعة التسلسل، المكررات)ـ

· الافتراضات: أن بيانات العدادات تتبع توزيعاً سلبياً ذا حدين، وأن التطبيع قد تم بشكل مناسب، وأن التصميم التجريبي يعكس السؤال العلميـ

· المخرجات: قائمة بالجينات ذات التعبير التفاضلي، مع قيم p معدلة وحجم التأثيرـ

· القيود: التحليل لا يأخذ في الاعتبار العوامل المخفية غير المقاسة إلا إذا أُضيفت إلى النموذج. كما أنه لا يحدد الآلية البيولوجيةـ

Applied example of DESeq2 analysis:

اختبار: قبل تشغيل DESeq2، تحقق من توزيع بيانات العدادات، وتأكد من أن العينات متجانسة، وأن التصميم مناسب. استخدم رسوم PCA و heatmaps كأدوات تشخيصية للكشف عن تأثيرات الدفعاتـ

· Inputs: a count matrix for each gene in each sample, and metadata about the samples (condition, sequencing batch, replicates).

· Outputs: a list of differentially expressed genes, with adjusted p-values and effect sizes.

· Limitations: the analysis does not account for unmeasured confounding factors unless added to the model. It also does not determine biological mechanism.

Test: before running DESeq2, check the distribution of count data, ensure samples are homogeneous, and that the design is appropriate. Use PCA and heatmap plots as diagnostic tools to detect batch effects.

عند تحليل بيانات تعبير، يجب تسجيل ما يلي:

· المصدر: البيانات الخام (ملفات FASTQ)، المرجع المستخدم، وقاعدة البيانات المستخدمة للتعليق التوضيحي.

· الإصدار: إصدار R، Bioconductor، الحزم المستخدمة (مثل DESeq2)، وإصدار المرجع.

· المعلمات: طريقة التطبيع، والعتبات المستخدمة (مثل قيمة p المعدلة، نسبة التغير).

· التحويلات: أي خطوات لتنقية البيانات أو تحويلها قبل التحليل (مثل فلترة الجينات منخفضة التعبير).

When analyzing expression data, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة التحليل والحصول على نفس النتائج دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: raw data files (FASTQ), reference used, and annotation database used.

· Version: R version, Bioconductor version, package versions (e.g., DESeq2), and reference version.

· Parameters: normalization method, and thresholds used (e.g., adjusted p-value, fold change).

· Transformations: any steps to clean or transform data before analysis (e.g., filtering low-expression genes).

قبل أن تحلل بيانات تعبير حقيقية، اختبر مسار العمل على بيانات اصطناعية ذات خصائص معروفة. أنشئ بيانات عدادات لجينات وهمية (مثلاً 100 جين، 12 عينة، حالتان، مع 3 مكررات لكل حالة). أضف اختلافاً بيولوجياً معروفاً (بعض الجينات تختلف بين الحالتين). طبّق نفس خطوات التحليل التي ستستخدمها مع البيانات الحقيقية (تطبيع، تحليل تعبير تفاضلي). تحقق من أن الأداة تكتشف الجينات المختلفة بدقة (حساسية ونوعية). إذا نجح الاختبار، فأنت

واثق من أن مسار العمل يعمل كما هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس خصائص البيانات وليس خطأ في الإعدادات.

Before analyzing real expression data, test the workflow on synthetic data with known properties. Create count data for hypothetical genes (e.g., 100 genes, 12 samples, two conditions, with 3 replicates per condition). Add known biological differences (some genes differ between conditions). Apply the same analysis steps you will use with real data (normalization, differential expression analysis). Verify that the tool accurately detects the different genes (sensitivity and specificity). If the test succeeds, the workflow is confident that the workflow works as expected, and any result with real data reflects the data's properties, not an error in the settings.

Part 7 — Biological Algorithms

الباب 7 — الخوارزميات الحيوية

الفكرة المركزية: exact search، approximate search، الفهرسة، التوسع.

Chapter 24 — Sequence Search

الفصل 24 — البحث في التسلسل

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Exact search answers a positional question: whether a specific pattern occurs and where. Approximate search adds a defined criterion for allowable differences. Finding a pattern is not the same as interpreting it; genomic and functional context determines whether the location matters.

البحث الدقيق يجيب عن سؤال موضعي: هل يظهر نمط بعينه، وأين؟ أما البحث التقريبي فيضيف معياراً للاختلاف ويحتاج إلى تعريف واضح لما يعنيه التطابق المقبول. العثور على نمط لا يساوي تفسيره؛ فالسياق الجيني والوظيفي هو الذي يحدد ما إذا كان الموقع مهماً.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Exact Search

1. البحث الدقيق (Exact Search)

Exact search is the simplest type of search: we look for a perfect (100%) match between a pattern and a text. In Python, we can use find() or in to search for a pattern in a string. This works well for small texts. But what if the text is a complete human genome (3 billion bases)? A linear search (scanning each character) would be extremely slow.

البحث الدقيق هو أبسط أنواع البحث: نبحث عن تطابق تام (100%) بين نمط (pattern) ونص (text). في Python، يمكننا استخدام find() أو in للبحث عن نمط في سلسلة نصية. هذا يعمل بشكل جيد للنصوص الصغيرة. لكن ماذا لو كان النص هو جينوم بشري كامل (3 مليارات قاعدة)؟ البحث الخطي (مسح كل حرف) سيكون بطيئاً جداً.

```
# بحث دقيق في Python
text = "ATGCGTACGTAGCTAGCTAGCTAGCTAGC"
pattern = "TAGC"
position = text.find(pattern)
if position != -1:
    print(f"النمط موجود عند الموقع {position}")
else:
    print("النمط غير موجود")
```

But exact search fails when there are even minor differences. In biological reality, sequences are rarely perfectly identical due to mutations, sequencing errors, or natural variation between species.

2. Approximate Search

لكن البحث الدقيق يفشل عندما يكون هناك اختلافات ولو بسيطة. في الواقع البيولوجي، التسلسلات نادراً ما تكون متطابقة تماماً بسبب الطفرات، أو أخطاء التسلسل، أو الاختلافات الطبيعية بين الأنواع.

Approximate search allows differences between the pattern and the text. These differences may be substitutions, insertions, or deletions. In bioinformatics, approximate search is the most commonly used, because it reflects biological reality. Tools like BLAST perform approximate search: they look for similar (not perfectly identical) regions between a query sequence and a reference database.

2. البحث التقريبي (Approximate Search)

البحث التقريبي يسمح بوجود اختلافات بين النمط والنص. هذه الاختلافات قد تكون تبدلات (substitutions)، إدخالات (insertions)، أو حذوفات (deletions). في المعلوماتية الحيوية، البحث التقريبي هو الأكثر استخداماً، لأنه يعكس الواقع البيولوجي. الأدوات مثل BLAST تقوم بالبحث التقريبي: تبحث عن مناطق مشابهة (وليس متطابقة تماماً) بين تسلسل استعلام (query) وقاعدة بيانات مرجعية.

مثال: البحث التقريبي

النمط: ATGCGTAC

النص: ATGCGTAC (تطابق تام) ✓

النص: ATGCGTTC (تبدل واحد) ✓ (تقريبي)

النص: ATGCGTA- (حذف واحد) ✓ (تقريبي)

النص: ATGCGTAAC (إدخال واحد) ✓ (تقريبي)

Approximate search is more complex than exact search, because it requires calculating a similarity score, not just a perfect match. The similarity score depends on substitution matrices (such as BLOSUM for proteins) and gap penalties.

البحث التقريبي أكثر تعقيداً من البحث الدقيق، لأنه يتطلب حساب درجة تشابه، وليس مجرد تطابق تام. درجة التشابه تعتمد على مصفوفات الاستبدال (مثل BLOSUM للبروتينات) ومعاملات الفجوات (gap penalties).

3. The Scale Problem: Searching in Large Databases

3. مشكلة الحجم: البحث في قواعد بيانات ضخمة

Direct search (scanning every character in every sequence) may be acceptable for a small file (a few hundred sequences). But it becomes impractical when dealing with huge databases like GenBank, which contains hundreds of millions of sequences. The solution is indexing.

البحث المباشر (مسح كل حرف في كل تسلسل) قد يكون مقبولاً لملف صغير (بضع مئات من التسلسلات). لكنه يصبح غير عملي عند التعامل مع قواعد بيانات ضخمة مثل GenBank، التي تحتوي على مئات الملايين من التسلسلات. الحل هو الفهرسة (Indexing) .

Indexing is building a data structure that allows fast searching. The most famous example is a Hash Table, where each pattern (or part of it) is stored with its location. When searching, we look up the table directly instead of scanning the entire text. In bioinformatics, specialized indexes such as Word Index, Suffix Tree, or Suffix Array are used to speed up search.

الفهرسة هي بناء هيكل بيانات يسمح بالبحث السريع. أشهر مثال هو جدول التجزئة (Hash Table) ، حيث يُخزَّن كل نمط (أو جزء منه) مع موقعه. عند البحث، نبحث في الجدول مباشرة بدلاً من مسح النص بأكمله. في المعلوماتية الحيوية، تُستخدم فهارس متخصصة مثل جدول الكلمات (Word Index) أو شجرة اللاحقات (Suffix Tree) أو مصفوفة اللاحقات (Suffix Array) لتسريع البحث.

مقارنة بين البحث المباشر والفهرسة:

البحث المباشر: $O(n * m)$ حيث n طول النص و m طول النمط

البحث بالفهرسة: $O(m)$ تقريباً (بحث في الجدول)

4. البحث في Biopython

4. Searching in Biopython

Biopython provides tools for searching for patterns in sequences, but it does not replace BLAST for large-scale approximate search. the workflow can use find(), count(), and startswith() on a Seq object. For approximate search, the workflow can use functions like Bio.SeqUtils.nt_search() to search for DNA patterns allowing a limited number of differences.

Biopython توفر أدوات للبحث عن الأنماط في التسلسلات، لكنها لا تحل محل BLAST للبحث التقريبي على نطاق واسع. يمكن استخدام find() و count() و startswith() على كائن Seq. للبحث التقريبي، يمكن استخدام دوال مثل Bio.SeqUtils.nt_search() للبحث عن أنماط DNA مع السماح بعدد محدود من الاختلافات.

```
from Bio.Seq import Seq
# بحث دقيق في Biopython
dna = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC")
pattern = "TAGC"
positions = [i for i in range(len(dna) - len(pattern) + 1) if dna[i:i+len(pattern)] == pattern]
print(f"المواقع: {positions}")
# بحث تقريبي في Biopython
def approximate_matches(sequence, pattern, max_mismatches=1):
  """تعليمي: يعيد مواضع التطابق التي لا تتجاوز عدد الاختلافات المحدد."""
  sequence = str(sequence)
  matches = []
  for i in range(len(sequence) - len(pattern) + 1):
    window = sequence[i:i + len(pattern)]
    mismatches = sum(a != b for a, b in zip(window, pattern))
    if mismatches <= max_mismatches:
      matches.append((i, mismatches))
  return matches
results = approximate_matches(dna, pattern, max_mismatches=1)
print(f"نتائج البحث التقريبي: {results}")
```

5. From Discovery to Interpretation

5. من الاكتشاف إلى التفسير

Sequence search tells you where the pattern is located, but it does not tell you why it is important. Finding a pattern at a specific location is just the beginning. Interpreting its significance requires context: is this pattern part of a known gene? Is it located in a coding or non-coding region? Is it conserved across species? These questions require annotations, databases, and independent evidence. Distinguishing between finding a pattern's location and interpreting its significance is the essence of critical thinking in bioinformatics.

البحث في التسلسل يخبرك أين يوجد النمط، لكنه لا يخبرك لماذا هو مهم. العثور على نمط في موقع معين هو مجرد بداية. تفسير أهميته يحتاج إلى سياق: هل هذا النمط جزء من جين معروف؟ هل يقع في منطقة مشفرة أم غير مشفرة؟ هل هو محفوظ عبر الأنواع؟ هذه الأسئلة تتطلب تعليقات توضيحية (annotations) وقواعد بيانات وأدلة مستقلة. الفصل بين اكتشاف موقع نمط وتفسير أهميته هو جوهر التفكير النقدي في المعلوماتية الحيوية.

Chapter Summary

خلاصة الفصل

Sequence search is a fundamental tool in bioinformatics. Exact search looks for a perfect match, while approximate search allows differences (substitutions, insertions, deletions). The scale problem makes direct search impractical for large databases, necessitating the use of indexing (such as hash tables, suffix trees). Biopython provides tools for small-scale exact and approximate search, but BLAST is the standard tool for large-scale approximate search. Most importantly, finding a pattern is just a first step; interpreting its significance requires context and additional evidence. The next chapter will delve into sequence distance measures: Hamming and Levenshtein.

البحث في التسلسل هو أداة أساسية في المعلوماتية الحيوية. البحث الدقيق يبحث عن تطابق تام، بينما البحث التقريبي يسمح باختلافات (تبدلات، إدخالات، حذوفات). مشكلة الحجم تجعل البحث المباشر غير عملي للقواعد الضخمة، مما يستدعي استخدام الفهارس (مثل جداول التجزئة، أشجار اللاحقات). Biopython توفر أدوات للبحث الدقيق والتقريبي على نطاق صغير، لكن BLAST هو الأداة القياسية للبحث التقريبي على نطاق واسع. الأهم من ذلك، العثور على نمط هو مجرد خطوة أولى؛ تفسير أهميته يحتاج إلى سياق وأدلة إضافية. الفصل القادم سيتعمق في مقاييس المسافة بين التسلسلات: Hamming و Levenshtein.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Sequence Search" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "البحث في التسلسل" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "finding a pattern in a sequence proves its biological significance." This is incorrect; finding a pattern is just a location discovery, not evidence of function. The correct statement is: "sequence search identifies pattern locations, but interpreting their biological significance requires context (such as annotations, evolutionary conservation, experimental evidence)."

A simple test: Practical check: "Does the mere presence of this pattern at this location tell me about its function?".

مثال على التجاوز: القول إن "العثور على نمط في تسلسل يثبت أهميته البيولوجية". هذا غير صحيح؛ العثور على نمط هو مجرد اكتشاف موقع، وليس دليلاً على الوظيفة. الصحيح هو: "البحث في التسلسل يحدد مواقع الأنماط، لكن تفسير أهميتها البيولوجية يتطلب سياقاً (مثل التعليقات التوضيحية، الحفظ التطوري، الأدلة التجريبية)." اختبار بسيط: فحص تطبيقي: "هل مجرد وجود هذا النمط في هذا الموقع يخبرني عن وظيفته؟"ـ

Example: Approximate search results giving multiple locations with 85% similarity is a computational representation. Claiming that "these locations represent homologous genes" is a biological claim not proven by similarity alone. Computational similarity prioritizes sites for further study, but it does not prove homology or function. Practical test: before inferring homology from computational similarity, look for additional evidence (such as gene structure, evolutionary conservation, experimental evidence).

مثال: نتيجة بحث تقريبي تعطي مواقع متعددة بتشابه 85% هي تمثيل حسابي. الادعاء بأن "هذه المواقع تمثل جينات متماثلة" هو ادعاء بيولوجي لا يثبته التشابه وحده. التشابه الحسابي يرشح مواقع لمزيد من الدراسة، لكنه لا يثبت التماثل أو الوظيفة. اختبار عملي: قبل أن تستنتج تماثلاً من تشابه حسابي، ابحث عن أدلة إضافية (مثل بنية الجين، الحفظ التطوري، الأدلة التجريبية)ـ

مثال تطبيقي على أداة BLAST:

· المدخلات: تسلسل استعلام (query)، قاعدة بيانات مرجعية، مصفوفة استبدال، عتبة E-valueـ

· الافتراضات: أن قاعدة البيانات ممثلة للتنوع البيولوجي، وأن المصفوفة مناسبة للمسافة التطورية، وأن عتبة E-value مضبوطة بشكل مناسبـ

· المخرجات: قائمة بالمطابقات المحتملة مع درجات التشابه وقيم الثقةـ

· القيود: BLAST قد يفوت مطابقات بعيدة التشابه، وقد يعطي إيجابيات كاذبة إذا كانت قاعدة البيانات غير مكتملة أو العتبة غير مضبوطةـ

Applied example of BLAST:

اختبار: قبل اعتماد نتائج BLAST، تحقق من E-value، وتوزيع المطابقات، وجودة المحاذاةـ

· Inputs: query sequence, reference database, substitution matrix, E-value threshold.

· Assumptions: that the database is representative of biological diversity, that the matrix is appropriate for the evolutionary distance, and that the E-value threshold is properly set.

· Outputs: a list of potential matches with similarity scores and confidence values.

· Limitations: BLAST may miss distantly related matches, and may produce false positives if the database is incomplete or the threshold is not properly set.

Test: before relying on BLAST results, check the E-value, match distribution, and alignment quality.

عند إجراء بحث في التسلسل، يجب تسجيل ما يلي:

· المصدر: التسلسل المستخدم (معرفه، مصدره)، وقاعدة البيانات المرجعية (إصدارها وتاريخها).

· الإصدار: إصدار الأداة المستخدمة (مثل BLAST 2.12.0)، وإصدار قاعدة البيانات.

· المعلمات: مصفوفة الاستبدال، ومعاملات الفجوات، وعتبة E-value، وأي معاملات أخرى.

· التحويلات: أي معالجة مسبقة للتسلسل قبل البحث (مثل قص المناطق منخفضة الجودة).

When performing a sequence search, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة البحث والحصول على نفس النتائج دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the sequence used (its identifier, source), and the reference database (its version and date).

· Version: the tool version used (e.g., BLAST 2.12.0), and the database version.

· Parameters: the substitution matrix, gap parameters, E-value threshold, and any other parameters.

· Transformations: any preprocessing of the sequence before search (e.g., trimming low-quality regions).

Test: Practical check: "If I gave this record to a colleague a year later, could they re-run the search and obtain the same results without my help?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على أداة بحث في التسلسل على بيانات حقيقية، اختبر الأداة على بيانات اصطناعية ذات مطابقات معروفة. أنشئ تسلسل استعلام، وقاعدة بيانات صغيرة تحتوي على مطابقات تامة وتقريبية معروفة. شغّل الأداة وتأكد من أنها تكتشف المطابقات المتوقعة (حساسية ونوعية). إذا نجح الاختبار، فأنت واثق من أن الأداة تعمل كما هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس العلاقات الحقيقية وليس خطأ في الإعدادات.

Before relying on a sequence search tool on real data, test the tool on synthetic data with known matches. Create a query sequence, and a small database containing known exact and approximate matches. Run the tool and verify that it detects the expected matches (sensitivity and specificity). If the test succeeds, the workflow is confident that the tool works as expected, and any result with real data reflects true relationships, not an error in the settings.

Chapter 25 — Hamming and Levenshtein Distance

الفصل 25 — Hamming و Levenshtein

الفكرة المركزية: مسافات السلاسل، deletions، insertions، substitutions، اختيار المقياس.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

Hamming وLevenshtein مثالان جيدان على أن تعريف المسافة جزء من تعريف المشكلة. Hamming يناسب السلاسل المتساوية الطول عندما يكون الاستبدال هو العملية المسموح بها، بينما Levenshtein يسمح بالإدراج والحذف والاستبدال. اختيار المقياس الخاطئ قد ينتج رقماً صحيحاً حسابياً لكنه لا يصف السؤال المطلوب.

Hamming and Levenshtein illustrate that defining a distance is part of defining the problem. Hamming fits equal-length strings when substitutions are the allowed operation, whereas Levenshtein allows insertions, deletions, and substitutions. A wrong metric can produce a computationally correct number that does not answer the intended question.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Are Distance Metrics؟

1. ما هي مقاييس المسافة؟

String distance metrics are mathematical functions that calculate the number of changes needed to transform one string into another. Each metric defines "change" differently: only substitutions? Are insertions and deletions allowed? How much does each operation cost? Choosing a metric is choosing a specific model of difference.

مقاييس المسافة بين السلاسل هي دوال رياضية تحسب عدد التغييرات اللازمة لتحويل سلسلة إلى أخرى. كل مقياس يعرّف "التغيير" بشكل مختلف: هل التبديل فقط؟ هل الإدراج والحذف مسموحان؟ كم تكلف كل عملية؟ اختيار المقياس هو اختيار لنموذج معين للاختلاف.

Distance metrics are not biological facts; they are models. There is no single "correct" metric, only a metric "appropriate" for a specific question. In biological sequences, insertions or deletions (indels) may be more common than substitutions in some contexts, and less common in others.

مقاييس المسافة ليست حقائق بيولوجية؛ إنها نماذج. لا يوجد مقياس "صحيح" دائماً، بل يوجد مقياس "مناسب" لسؤال معين. في التسلسلات البيولوجية، قد يكون الإدراج أو الحذف (indels) أكثر شيوعاً من التبديل في بعض السياقات، وأقل شيوعاً في سياقات أخرى.

2. مسافة Hamming

2. Hamming Distance

Hamming distance is the simplest distance metric. It counts the number of positions where the letters differ between two strings of equal length. It does not allow insertions or deletions. If the two strings differ in length, Hamming distance cannot be calculated.

مسافة Hamming هي أبسط مقاييس المسافة. تحسب عدد المواضع التي تختلف فيها الحروف بين سلسلتين متساويتين في الطول. لا تسمح بالإدراجات أو الحذوفات. إذا كان السلسلتان مختلفتين في الطول، لا يمكن حساب مسافة Hamming.

def hamming_distance(s1, s2):

"""حساب مسافة Hamming بين سلسلتين متساويتين في الطول."""

if len(s1) != len(s2):

raise ValueError("السلسلتان يجب أن تكونا متساويتين في الطول")

```
    return sum(c1 != c2 for c1, c2 in zip(s1, s2))
# مثال
s1 = "ATGC"
s2 = "ATGG"
print(hamming_distance(s1, s2)) # الناتج: 1 (الموضع الأخير مختلف)
s3 = "ATGC"
s4 = "ATGA"
print(hamming_distance(s3, s4)) # الناتج: 1 (الموضع الأخير مختلف)
s5 = "AAAA"
s6 = "TTTT"
print(hamming_distance(s5, s6)) # الناتج: 4 (جميع المواضع مختلفة)
```

الدوال المعرفة هنا (hamming_distance) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة.

القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها.

Hamming distance is suitable for contexts where we expect sequences to have the same length, and differences are only substitutions. For example, comparing short sequences of the same length, or comparing codons (triplets).

Limitations of Hamming: It does not handle insertions or deletions. If there is a single insertion or deletion, the distance will fail to represent the true similarity, because the lengths will differ.

مسافة Hamming مناسبة للسياقات التي نتوقع فيها أن تكون التسلسلات متساوية الطول، والاختلافات عبارة عن تبديلات فقط. مثلاً، مقارنة تسلسلات قصيرة من نفس الطول، أو مقارنة كودونات (ثلاثية القواعد).

3. Levenshtein Distance (Edit Distance)

حدود Hamming: لا تتعامل مع الإدراجات أو الحذوفات. إذا كان هناك إدراج أو حذف واحد، ستفشل المسافة في تمثيل التشابه الحقيقي، لأن طول السلسلتين سيختلف.

Levenshtein distance, or Edit Distance, calculates the minimum number of operations needed to transform one string into another. The allowed operations are: substitution, insertion, and deletion. Each operation typically costs 1, but the cost can be adjusted (e.g., making substitution cost 2 and insertion cost 1). Levenshtein distance is more flexible than Hamming because it handles strings of different lengths.

3. مسافة Levenshtein (المسافة التحريرية)

مسافة Levenshtein، أو المسافة التحريرية (Edit Distance)، تحسب أقل عدد من العمليات اللازمة لتحويل سلسلة إلى أخرى. العمليات المسموح بها هي: التبديل (substitution)، الإدراج (insertion)، والحذف (deletion). كل عملية تكلف عادةً 1، لكن يمكن تعديل التكلفة (مثل جعل تكلفة التبديل 2 والإدراج 1). مسافة Levenshtein أكثر مرونة من Hamming، لأنها تتعامل مع السلاسل مختلفة الطول.

```
    if len(s2) == 0:
        return len(s1)
def levenshtein_distance(s1, s2):
    """حساب مسافة Levenshtein (خوارزمية البرمجة الديناميكية) بين سلسلتين."""
    if len(s1) < len(s2):
```

return levenshtein_distance(s2, s1)
previous_row = range(len(s2) + 1)
for i, c1 in enumerate(s1):
current_row = [i + 1]
for j, c2 in enumerate(s2):
insertions = previous_row[j + 1] + 1
deletions = current_row[j] + 1
substitutions = previous_row[j] + (c1 != c2)

```
        current_row.append(min(insertions, deletions, substitutions))
    previous_row = current_row
return previous_row[-1]
# مثال
s1 = "ATGC"
s2 = "ATGG"
print(levenshtein_distance(s1, s2)) # 1 :الناتج
s1 = "ATGC"
s2 = "ATG"  # حذف قاعدة
print(levenshtein_distance(s1, s2)) # 1 :الناتج (حذف واحد)
s1 = "ATGC"
s2 = "ATGCGT" # إدراج قاعدتين
print(levenshtein_distance(s1, s2)) # 2 :الناتج (إدراجان)
s1 = "ACTG"
s2 = "ACGT" # تبديل C و G
```

print(levenshtein_distance(s1, s2)) # الناتج: 2 (تبديل + إدراج/حذف أو تبديلان حسب المحاذاة)

الدوال المعرفة هنا (levenshtein_distance) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة.

القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

Levenshtein distance is suitable for contexts where insertions and deletions may occur, such as comparing DNA sequences from different organisms, or comparing protein sequences where gaps are common.

4. Comparison Between Hamming and Levenshtein

مسافة Levenshtein مناسبة للسياقات التي قد تحدث فيها إدراجات وحذوفات، مثل مقارنة تسلسلات DNA من كائنات مختلفة، أو مقارنة تسلسلات بروتين حيث تكون الفجوات شائعة.

الميزة Hamming Levenshtein

4. مقارنة بين Levenshtein و Hamming

العمليات المسموح بها تبديل فقط تبديل، إدراج، حذف.

أطوال السلاسل يجب أن تكون متساوية يمكن أن تكون مختلفة.

Feature Hamming Levenshtein

التعقيد الحسابي O(n) (خطي) O(n*m) (تربيعي).

Allowed operations Substitution only Substitution, insertion, deletion.

الاستخدام النموذجي مقارنات قصيرة، متساوية الطول مقارنات عامة، متغيرة الطول.

String lengths Must be equal Can be different.

الحساسية للفجوات لا تتعامل مع الفجوات تتعامل مع الفجوات.

Computational complexity O(n) (linear) O(n*m) (quadratic).

Typical use Short, equal-length comparisons General, variable-length comparisons.

Gap handling Does not handle gaps Handles gaps.

5. Choosing the Right Measure

Choosing the measure should follow the nature of the data and the biological question, not be a programming habit:

5. اختيار المقياس المناسب

اختيار المقياس يجب أن يتبع طبيعة البيانات والسؤال البيولوجي، لا أن يكون عادة برمجية:

· If sequences are equal in length and changes are only substitutions (e.g., comparing SNP sites at the same position): use Hamming.

· إذا كانت التسلسلات متساوية الطول، والتغييرات هي تبديلات فقط (مثل مقارنة مواقع SNP في نفس الموضع):

استخدم Hamming.

· If sequences may contain insertions or deletions (e.g., comparing genes from different organisms, or comparing protein sequences): use Levenshtein or sequence alignment (e.g., Needleman-Wunsch or Smith-Waterman) that allow different gap costs.

· إذا كانت التسلسلات قد تحتوي على إدراجات أو حذوفات (مثل مقارنة جينات من كائنات مختلفة، أو مقارنة

تسلسلات البروتين): استخدم Levenshtein أو محاذاة التسلسل (مثل Needleman-Wunsch أو Smith-

Waterman) التي تسمح بتكاليف مختلفة للفجوات.

· If the evolutionary distance is large, you may need more complex models (such as nucleotide or amino acid substitution models) that account for the varying frequencies of different substitutions.

· إذا كانت المسافة التطورية بعيدة، قد تحتاج إلى نماذج أكثر تعقيداً (مثل نماذج استبدال النيوكليوتيدات أو الأحماض الأمينية) التي تأخذ في الاعتبار تواتر التبديلات المختلفة.

6. Practical Application: Comparing Sequences Using Distances

Let's apply what we have learned to a small set of sequences.

6. التطبيق العملي: مقارنة تسلسلات باستخدام المسافات

دعنا نطبق ما تعلمناه على مجموعة صغيرة من التسلسلات.

```
# تسلسلات عينة
seq1 = "ATGCGTAC"
seq2 = "ATGCGTTC" # تبديل واحد
seq3 = "ATGCGTA-" # حذف واحد (إذا عُدل بالرمز -)
seq4 = "ATGCGTAAC" # إدراج واحد
try:
    print(f"seq1 vs seq3: {hamming_distance(seq1, seq3)}")
except ValueError as e:
    print(f"seq1 vs seq3: خطأ - {e} (أطوال مختلفة)")
# حساب المسافات (مع التنبيه على حالة seq3 التي تختلف طولاً)
print("Hamming distances (بين seq1 والأخرى):")
try:
    print(f"seq1 vs seq2: {hamming_distance(seq1, seq2)}")
except ValueError as e:
    print(f"seq1 vs seq2: خطأ - {e}")
try:
    print(f"seq1 vs seq4: {hamming_distance(seq1, seq4)}")
except ValueError as e:
    print(f"seq1 vs seq4: خطأ - {e} (أطوال مختلفة)")
print("\nLevenshtein distances:")
print(f"seq1 vs seq2: {levenshtein_distance(seq1, seq2)}")  # 1
print(f"seq1 vs seq3: {levenshtein_distance(seq1, seq3)}")  # 1 (حذف)
print(f"seq1 vs seq4: {levenshtein_distance(seq1, seq4)}")  # 1 (إدراج)
```

This example shows the practical difference: Hamming fails with different lengths, while Levenshtein handles them flexibly.

هذا المثال يوضح الفرق العملي: Hamming تفشل مع الأطوال المختلفة، بينما Levenshtein تتعامل معها بمرونة.

7. Gap Assumptions

7. افتراضات الفجوات (Gap Assumptions)

Distance is not just a number; it is a definition of what we consider a change. In Levenshtein, each operation has the same cost (1). But in biological reality, inserting or deleting a base may have a different cost than substituting. In advanced sequence alignment (such as BLAST, or global and local alignment algorithms), different gap costs (gap opening penalty and gap extension penalty)

are used to reflect the fact that long gaps are often less costly than several short gaps.

المسافة ليست مجرد عدد؛ إنها تعريف لما نعتبره تغييراً. في Levenshtein، كل عملية لها نفس التكلفة (1). لكن في الواقع البيولوجي، إدراج أو حذف قاعدة قد يحمل كلفة مختلفة عن التبديل. في محاذاة التسلسل المتقدمة (مثل BLAST، أو خوارزميات المحاذاة العالمية والمحلية)، تُستخدم تكاليف مختلفة للفجوات (gap opening penalty و gap extension penalty) لتعكس حقيقة أن الفجوات الطويلة غالباً ما تكون أقل تكلفة من عدة فجوات قصيرة.

مثال على تكاليف مختلفة:

تبديل: 1

إدراج: 2

حذف: 2

فتح فجوة: 5

تمديد فجوة: 1

Choosing these costs affects the alignment result and the calculated distance. Therefore, one must know the assumptions underlying any metric or algorithm before using it.

اختيار هذه التكاليف يؤثر على نتيجة المحاذاة والمسافة المحسوبة. لذلك، يجب معرفة الافتراضات التي يقوم عليها أي مقياس أو خوارزمية قبل استخدامها.

Chapter Summary

خلاصة الفصل

Hamming distance and Levenshtein distance are two fundamental string distance metrics. Hamming is simple but requires equal lengths and only allows substitutions. Levenshtein is more flexible, allowing insertion and deletion, but is more computationally complex. Choosing the appropriate measure depends on the nature of the data and the biological question. Distance is not just a number; it is a model carrying assumptions about how changes occur. Understanding these assumptions is what prevents misinterpretation of results. The next chapter will move to computational complexity, to understand how data size affects algorithm selection.

مسافة Hamming ومسافة Levenshtein هما مقياسان أساسيان للمسافة بين السلاسل. Hamming بسيط ولكنه يتطلب أطوالاً متساوية ولا يسمح إلا بالتبديلات. Levenshtein أكثر مرونة، ويسمح بالإدراج والحذف، لكنه أكثر تعقيداً حسابياً. اختيار المقياس المناسب يعتمد على طبيعة البيانات والسؤال البيولوجي. المسافة ليست مجرد رقم؛ هي نموذج يحمل افتراضات حول كيفية حدوث التغييرات. فهم هذه الافتراضات هو ما يمنع سوء تفسير النتائج. الفصل القادم سينتقل إلى التعقيد الحسابي، لفهم كيف تؤثر حجم البيانات على اختيار الخوارزميات.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Hamming and Levenshtein Distance" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Hamming و Levenshtein" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "Hamming distance is always the best measure because it is simpler." This is incorrect; simplicity is not a criterion for correctness. The correct statement is: "Hamming and Levenshtein are different models for measuring difference, and choosing between them depends on whether insertions and deletions are expected in the data." A simple test: Practical check: "Do the sequences I am comparing have the same length, and are differences only substitutions?".

مثال على التجاوز: القول إن "مسافة Hamming هي المقياس الأفضل دائماً لأنها أبسط". هذا غير صحيح؛ البساطة ليست معياراً للصواب. الصحيح هو: "Hamming و Levenshtein هما نموذجان مختلفان لقياس الاختلاف، واختيار أيهما يعتمد على ما إذا كانت الإدراجات والحذوفات متوقعة في البيانات." اختبار بسيط: فحص تطبيقي: "هل التسلسلات التي أقارنها لها نفس الطول، وهل الاختلافات هي تبديلات فقط؟".

Example: A Levenshtein distance of 3 between two sequences is a computational representation. Claiming that "these two sequences are functionally different because their distance is 3" is a biological claim not justified by the distance alone. The distance tells us about the number of edit operations, but it does not tell us about the effect on function. Practical test: before inferring a functional effect from a computational distance, check the location of changes (are they in a functional region?) and evolutionary conservation.

مثال: مسافة Levenshtein بين تسلسلين تساوي 3 هي تمثيل حسابي. الادعاء بأن "هذين التسلسلين مختلفان وظيفياً لأن مسافتهما 3" هو ادعاء بيولوجي غير مبرر بالمسافة وحدها. المسافة تخبرنا عن عدد العمليات التحريرية،

لكنها لا تخبرنا عن التأثير على الوظيفة. اختبار عملي: قبل أن تستنتج تأثيراً وظيفياً من مسافة حسابية، تحقق من موقع التغييرات (هل هي في منطقة وظيفية؟) ومن الحفظ التطوري.

مثال تطبيقي على مسافة Levenshtein:

· المدخلات: سلسلتان نصيتان (قد تكونان مختلفتين في الطول)، وتكاليف العمليات (افتراضياً كلها 1).

· الافتراضات: أن كل عملية (تبديل، إدراج، حذف) لها نفس التكلفة، وأن الترتيب النسبي للرموز مهم.

· المخرجات: عدد صحيح يمثل أقل عدد من العمليات لتحويل سلسلة إلى أخرى.

· القيود: لا تأخذ في الاعتبار التشابه الكيميائي بين الأحماض الأمينية (مثل أن استبدال الحمض الأميني بمشابه كيميائياً قد يكون أقل تأثيراً)، ولا تأخذ في الاعتبار السياق البيولوجي.

Applied example of Levenshtein distance:

اختبار: قبل استخدام أي مقياس مسافة، تحقق من تكافؤ التكاليف مع السياق البيولوجي. إذا كانت الإدراجات أكثر شيوعاً، قد تحتاج إلى تكلفة أقل للإدراج.

· Inputs: two strings (may differ in length), and operation costs (default all 1).

· Assumptions: that each operation (substitution, insertion, deletion) has the same cost, and that the relative order of symbols matters.

· Outputs: an integer representing the minimum number of operations to transform one string into another.

· Limitations: it does not account for chemical similarity between amino acids (e.g., substituting an amino acid with a chemically similar one may have less effect), nor does it account for biological context.

Test: before using any distance measure, check whether the costs are equivalent to the biological context. If insertions are more common, you may need a lower cost for insertion.

عند استخدام مقاييس المسافة، يجب تسجيل ما يلي:

· الإصدار: إصدار أي مكتبة استخدمتها (مثل textdistance، أو إذا كتبت الكود بنفسك).

· المعلمات: تكاليف العمليات (إذا كانت مختلفة عن الافتراضية).

· التحويلات: أي معالجة مسبقة للتسلسلات قبل حساب المسافة (مثل إزالة الفجوات، أو تحويل الحروف إلى أحرف كبيرة).

When using distance measures, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة حساب المسافات والحصول على نفس النتائج دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Version: the version of any library used (e.g., textdistance, or if you wrote the code the analysis itself).

· Parameters: operation costs (if different from default).

· Transformations: any preprocessing of sequences before distance calculation (e.g., removing gaps, converting to uppercase).

Test: Practical check: "If I gave this record to a colleague a year later, could they re-calculate the distances and obtain the same results without my help?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على مقياس مسافة في تحليل بيانات حقيقية، اختبره على تسلسلات اصطناعية ذات اختلافات معروفة. أنشئ تسلسلاً أصلياً، ثم أنشئ نسخاً منه تحتوي على تبديل واحد، إدراج واحد، حذف واحد، ومجموعات منها. احسب المسافات بين التسلسل الأصلي والنسخ، وتأكد من أن المسافات تعكس الاختلافات المتوقعة. إذا نجح الاختبار، فأنت واثق من أن المقياس يعمل كما هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس العلاقات الحقيقية وليس خطأ في المقياس.

Before relying on a distance measure in real data analysis, test it on synthetic sequences with known differences. Create an original sequence, then create copies containing one substitution, one insertion, one deletion, and combinations of them. Calculate the distances between the original and the copies, and verify that the distances reflect the expected differences. If the test succeeds, the workflow is confident that the measure works as expected, and any result with real data reflects true relationships, not an error in the measure.

Chapter 26 — Computational Complexity

الفصل 26 — التعقيد الحسابي

الفكرة المركزية: Big-O، الزمن، الذاكرة، قابلية التوسع، المفاضلة.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

التعقيد يصف كيف يتغير استهلاك الموارد مع حجم المدخلات. Big-O ليس ساعة توقيت؛ فهو يصف معدل النمو ويهمل عوامل ثابتة وتفاصيل العتاد. لذلك يجب التمييز بين التعقيد النظري والزمن الفعلي، مع الانتباه إلى الذاكرة، وحجم البيانات، والتوازي، وخصائص المدخلات.

Complexity describes how resource use grows with input size. Big-O is not a stopwatch; it describes growth rate while abstracting away constants and hardware details. Theoretical complexity should therefore be distinguished from measured runtime, memory use, parallelism, and input characteristics.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

إنفوجرافيك 8 / *Infographic 8*

1. What Is Computational Complexity؟

1. ما هو التعقيد الحسابي؟

Computational complexity is the study of how an algorithm's time and memory requirements change as the input size grows. Complexity is usually expressed using Big-O notation, which describes the asymptotic behavior of the algorithm. Big-O does not give you a specific time in seconds; it gives you an idea of how runtime or memory usage grows as the input size grows.

التعقيد الحسابي هو دراسة كيفية تغير متطلبات الزمن والذاكرة للخوارزمية مع تغير حجم المدخلات. يُعبَّر عن التعقيد عادةً باستخدام تدوين Big-O، الذي يصف السلوك المقارب (asymptotic behavior) للخوارزمية. Big-O لا يعطيك وقتاً محدداً بالثواني؛ بل يعطيك فكرة عن كيفية نمو زمن التشغيل أو استخدام الذاكرة عندما يكبر حجم المدخلات.

Big-O: النمو المقارب

Big-O تصف أسوأ حالة (worst-case) لنمو الخوارزمية. بعض التعقيدات الشائعة:

· O(1): زمن ثابت (independent of input size). مثلاً: الوصول إلى عنصر في مصفوفة باستخدام الفهرس.

· O(log n): زمن لوغاريتمي. مثلاً: البحث في شجرة ثنائية متوازنة.

· O(n): زمن خطي. مثلاً: البحث عن عنصر في قائمة غير مرتبة (مسح كل العناصر).

· O(n log n): زمن شبه خطي. مثلاً: خوارزميات الفرز الفعالة (مثل Merge Sort، Quick Sort).

· $O(n^2)$: زمن تربيعي. مثلاً: مقارنة كل عنصر مع كل عنصر آخر (مثل مسافة Levenshtein بين سلسلتين طويلتين، أو BLAST في أسوأ حالاته بدون فهرسة).

Big-O: Asymptotic Growth

· $O(n^3)$: زمن تكعيبي. نادر في المعلوماتية الحيوية، لكن قد يظهر في خوارزميات معينة.

Big-O describes the worst-case growth of an algorithm. Some common complexities:

· $O(2^n)$: زمن أسي. غير قابل للتطبيق عملياً إلا على مدخلات صغيرة جداً (مثل بعض خوارزميات البرمجة الديناميكية للتركيب البروتيني).

· O(1): constant time (independent of input size). Example: accessing an element in an array by index.

· O(log n): logarithmic time. Example: searching in a balanced binary tree.

· O(n): linear time. Example: searching for an element in an unsorted list (scanning all elements).

· O(n log n): linearithmic time. Example: efficient sorting algorithms (e.g., Merge Sort, Quick Sort).

· $O(n^2)$: quadratic time. Example: comparing every element with every other element (e.g., Levenshtein distance between two long sequences, or BLAST in worst case without indexing).

· $O(n^3)$: cubic time. Rare in bioinformatics, but may appear in certain algorithms.

· $O(2^n)$: exponential time. Practically infeasible except on very small inputs (e.g., some dynamic programming algorithms for protein folding).

Language / نوع الكود: **Text / Complexity Example**

مقارنة نمو التعقيدات مع زيادة حجم المدخلات (n):

$n = 10$: $O(n) = 10$, $O(n^2) = 100$, $O(2^n) = 1024$

$n = 100$: $O(n) = 100$, $O(n^2) = 10000$, $O(2^n) = 1.27e30$ (غير قابل للتطبيق)

$n = 1000$: $O(n) = 1000$, $O(n^2) = 1e6$, $O(2^n)$ = غير قابل للتطبيق تماماً

2. Why Is Computational Complexity Important in Bioinformatics؟

2. لماذا التعقيد الحسابي مهم في المعلوماتية الحيوية؟

Biological data is unimaginably large. A single human genome contains 3 billion base pairs. The GenBank database contains hundreds of millions of sequences. A single RNA-seq project may produce hundreds of millions of reads. If you do not consider computational complexity, you may discover that the project's analysis, which took weeks to write, takes months to run, or consumes more memory than available.

البيانات البيولوجية ضخمة بشكل لا يمكن تصوره. جينوم بشري واحد يحتوي على 3 مليارات زوج قاعدي. قاعدة بيانات GenBank تحتوي على مئات الملايين من التسلسلات. مشروع واحد من RNA-seq قد ينتج مئات الملايين من القراءات. إذا لم تأخذ التعقيد الحسابي في الاعتبار، فقد تكتشف أن تحليلك الذي استغرق أسابيع في الكتابة، يستغرق شهوراً في التشغيل، أو يستهلك ذاكرة أكثر مما يتوفر.

أمثلة من المعلوماتية الحيوية:

مثال 1: مقارنة كل زوج من التسلسلات.

· يوجد في المثال N تسلسلاً، تريد مقارنة كل زوج. عدد المقارنات = $O(N^2) \approx N*(N-1)/2$.

· إذا كان $N = 1000$، فالمقارنات ≈ 500,000.

· إذا كان $N = 1,000,000$، فالمقارنات ≈ 5×10^{11} (غير قابل للتطبيق).

· الحل: استخدام الفهرسة (مثل BLAST) لتقليل المقارنات إلى $O(N \log N)$ أو أقل.

Example 2: محاذاة تسلسلين باستخدام البرمجة الديناميكية (Needleman-Wunsch).

· التعقيد: $O(n*m)$ حيث n و m طولا التسلسلين.

· إذا كان $n = m = 1000$، فـ $O(1,000,000)$ (مليون عملية) — مقبول.

· إذا كان $n = m = 100,000$، فـ $O(10^{10})$ (10 مليار عملية) — غير مقبول.

مثال 3: البحث في قاعدة بيانات ضخمة.

· البحث الخطي (مسح كل تسلسل) = $O(N * L)$ حيث N عدد التسلسلات و L متوسط الطول.

3. Time vs. Memory

· استخدام فهرس (مثل جدول تجزئة للـ k-mers) = $O(L)$ تقريباً.

There is often a trade-off between time and memory. the workflow can speed up an algorithm by using more memory (such as storing intermediate results, or building indexes). Or the workflow can reduce memory usage at the cost of increased time (such as processing data as a stream instead of loading it all into memory). the project's choice depends on what is available: do the workflow has enough memory? Is time the limiting factor?

3. الزمن مقابل الذاكرة (Time vs. Memory)

غالباً ما تكون هناك مفاضلة (trade-off) بين الزمن والذاكرة. يمكن تسريع خوارزمية باستخدام ذاكرة أكثر (مثل تخزين نتائج وسيطة، أو بناء فهارس). أو يمكن تقليل استخدام الذاكرة على حساب زيادة الزمن (مثل معالجة البيانات على شكل تدفق بدلاً من تحميلها كلها في الذاكرة). اختيارك يعتمد على ما هو متاح: هل يوجد في المثال ذاكرة كافية؟ هل الوقت هو العامل المحدد؟

مثال على المفاضلة:

خوارزمية A: $O(n)$ زمن، $O(1)$ ذاكرة (سريعة، قليلة الذاكرة) ✓

خوارزمية B: $O(\log n)$ زمن، $O(n)$ ذاكرة (أسرع، لكن تستهلك ذاكرة أكثر)

In practice, memory is often the limiting factor in bioinformatics, especially when dealing with whole-genome data. But with increasing memory availability, the trade-off may shift towards using more memory to reduce time.

4. Indexing as a Solution to the Scale Problem

في الممارسة العملية، غالباً ما تكون الذاكرة هي العامل المحدد في المعلوماتية الحيوية، خاصة عند التعامل مع بيانات الجينوم الكاملة. لكن مع زيادة توفر الذاكرة، قد تتحول المفاضلة نحو استخدام المزيد من الذاكرة لتقليل الزمن.

Indexing is a technique for building a data structure that allows fast searching, at the cost of an initial overhead (building the index) and additional memory. In bioinformatics, indexing is essential for handling large data. The most famous example is BLAST, which uses a word index to speed up search. Instead of comparing each query sequence with every sequence in the database, BLAST first searches for common words, then extends matches.

4. الفهرسة كحل لمشكلة الحجم

الفهرسة هي تقنية لبناء هيكل بيانات يسمح بالبحث السريع، مقابل تكلفة أولية (بناء الفهرس) وذاكرة إضافية. في المعلوماتية الحيوية، الفهرسة أساسية للتعامل مع البيانات الضخمة. أشهر مثال هو BLAST، الذي يستخدم فهرساً للكلمات (words) لتسريع البحث. بدلاً من مقارنة كل تسلسل استعلام مع كل تسلسل في قاعدة البيانات، يبحث BLAST أولاً عن كلمات مشتركة، ثم يوسع المطابقات.

فكرة الفهرسة في BLAST:

1. بناء جدول لجميع الكلمات (مثل كل 3 قواعد) في قاعدة البيانات.
2. عند البحث، استخرج كل الكلمات من تسلسل الاستعلام.
3. ابحث عن الكلمات في الجدول ($O(1)$ تقريباً).
4. قم بتوسيع المطابقات التي تم العثور عليها.

5. Streaming and Batching

When data is too large to be loaded into memory all at once, the workflow can use streaming or batching. Streaming means reading data in parts, processing it as you read, without loading it all into memory. Batching means processing data in groups instead of all at once. These techniques allow processing huge data on machines with limited memory.

5. المعالجة المتدفقة (Streaming) والتجميع (Batching)

عندما تكون البيانات كبيرة جداً بحيث لا يمكن تحميلها في الذاكرة دفعة واحدة، يمكن استخدام المعالجة المتدفقة (streaming) أو التجميع (batching). المعالجة المتدفقة تعني قراءة البيانات جزءاً جزءاً، ومعالجتها أثناء القراءة،

دون تحميلها كلها في الذاكرة. التجميع يعني معالجة البيانات في مجموعات (batches) بدلاً من كل البيانات دفعة واحدة. هذه التقنيات تسمح بمعالجة بيانات ضخمة على أجهزة ذات ذاكرة محدودة.

```
# مثال: معالجة ملف FASTA كبير سطراً بسطر باستخدام SeqIO.parse
from Bio import SeqIO
# هذا لا يحمل الملف كله في الذاكرة
for record in SeqIO.parse("large_file.fasta", "fasta"):
    # معالجة كل سجل على حدة
    print(record.id, len(record.seq))
```

6. Practical Application: Estimating Complexity Before Implementation

6. التطبيق العملي: تقدير التعقيد قبل التنفيذ

قبل أن تكتب كوداً لتحليل بيانات كبيرة، قم بتقدير التعقيد:

· كم عدد العناصر (N)؟ (عدد التسلسلات، عدد القراءات، عدد الجينات).

· ما هو متوسط حجم كل عنصر؟ (متوسط طول التسلسل).

Before writing code to analyze large data, estimate the complexity:

· ما هو التعقيد الزمني للخوارزمية التي تخطط لاستخدامها؟ (O(n)، O(n^2)، إلخ).

· How many elements (N)? (number of sequences, number of reads, number of genes).

· كم من الذاكرة ستستهلك في أسوأ حالة؟

· What is the average size of each element? (average sequence length).

· هل يمكن استخدام فهرسة، أو معالجة متدفقة، أو تجميع لتقليل التعقيد؟

· What is the time complexity of the algorithm you plan to use? (O(n), O(n^2), etc.).

· How much memory will it consume in the worst case?

· Can you use indexing, streaming, or batching to reduce complexity?

مثال: تقدير لمشروع RNA-seq

- عدد القراءات: 50 مليون

- متوسط طول القراءة: 150 قاعدة

- المحاذاة باستخدام خوارزمية O(n*m): غير ممكن (7.5 = 150 * 50M مليار عملية)

- المحاذاة باستخدام فهرس (مثل Bowtie2): O(n log n) أو أفضل، ممكن

- الذاكرة: قد تحتاج إلى 8-16 جيجابايت للفهرس

7. Limits of Computational Complexity in Practice

7. حدود التعقيد الحسابي في الممارسة

التعقيد الحسابي ليس العامل الوحيد. في الممارسة العملية، هناك عوامل أخرى تؤثر على الأداء:

· ثوابت الخوارزمية (Constants): خوارزمية O(n) مع ثابت كبير قد تكون أبطأ من خوارزمية O(n log n) مع ثابت صغير على مدخلات صغيرة.

Computational complexity is not the only factor. In practice, other factors affect performance:

· البنية التحتية (Infrastructure): سرعة المعالج، حجم الذاكرة، سرعة القرص الصلب، التخزين المؤقت (caching).

· Algorithm constants: An O(n) algorithm with a large constant may be slower than an O(n log n) algorithm with a small constant on small inputs.

· توازي المعالجة (Parallelism): بعض الخوارزميات يمكن تنفيذها بشكل متوازٍ على عدة معالجات، مما يقلل الزمن الفعلي.

· Infrastructure: processor speed, memory size, hard disk speed, caching.

· خصائص البيانات: خوارزمية تكون سريعة في المتوسط لكن بطيئة في أسوأ الحالات قد تكون مقبولة إذا كانت بياناتك نموذجية.

· Parallelism: some algorithms can be executed in parallel on multiple processors, reducing actual time.

· Data properties: An algorithm that is fast on average but slow in the worst case may be acceptable if the project's data is typical.

Chapter Summary

خلاصة الفصل

Computational complexity is the study of how time and memory change with input size. Big-O notation describes the asymptotic growth of algorithms. Biological data is large, making computational complexity a critical factor in algorithm selection and workflow design. Indexing, streaming, and batching are techniques for handling large data. There is a trade-off between time and memory, and the choice depends on available resources. Estimating complexity before implementation is a necessary practice in bioinformatics. The next chapter will move to Clustering, a method for exploring data structure and grouping similar elements.

التعقيد الحسابي هو دراسة كيفية تغير الزمن والذاكرة مع حجم المدخلات. تدوين Big-O يصف النمو المقارب للخوارزميات. البيانات البيولوجية ضخمة، مما يجعل التعقيد الحسابي عاملاً حاسماً في اختيار الخوارزميات وتصميم خطوط العمل. الفهرسة، والمعالجة المتدفقة، والتجميع هي تقنيات للتعامل مع البيانات الضخمة. هناك مفاضلة بين الزمن والذاكرة، ويعتمد الاختيار على الموارد المتاحة. تقدير التعقيد قبل التنفيذ هو ممارسة ضرورية في المعلوماتية الحيوية. الفصل القادم سينتقل إلى التجميع (Clustering)، وهو طريقة لاستكشاف بنية البيانات وتجميع العناصر المتشابهة.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Computational Complexity" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "التعقيد الحسابي" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "an O(n) algorithm is always better than $O(n^2)$." This is not always true; constants and infrastructure matter. The correct statement is: "an algorithm with lower theoretical complexity is usually better for large data, but practical factors (such as constants, available memory, and infrastructure) may affect actual performance." A simple test: Practical check: "Have I compared theoretical complexity with the hardware constraints I will use?".

مثال على التجاوز: القول إن "خوارزمية $O(n)$ دائماً أفضل من $O(n^2)$". هذا ليس صحيحاً دائماً؛ فالثوابت والبنية التحتية تلعب دوراً. الصحيح هو: "خوارزمية ذات تعقيد أقل نظرياً تكون عادةً أفضل للبيانات الكبيرة، لكن العوامل العملية (مثل الثوابت، والذاكرة المتاحة، والبنية التحتية) قد تؤثر على الأداء الفعلي." اختبار بسيط: فحص تطبيقي: "هل قارنت التعقيد النظري مع قيود الأجهزة التي سأستخدمها؟".

Example: Choosing an O(n log n) algorithm instead of $O(n^2)$ for data analysis is a computational decision. Claiming that "this analysis reflects biological truth" is a biological claim not proven by algorithm choice. The algorithm improves performance, but it does not prove biological hypotheses. Practical test: before confusing computational efficiency with biological validity, ensure that statistical and experimental evidence supports the project’s conclusions.

مثال: اختيار خوارزمية $O(n \log n)$ بدلاً من $O(n^2)$ لتحليل البيانات هو قرار حسابي. الادعاء بأن "هذا التحليل يعكس الحقيقة البيولوجية" هو ادعاء بيولوجي لا يثبته اختيار الخوارزمية. الخوارزمية تحسن الأداء، لكنها لا تثبت صحة الفرضيات البيولوجية. اختبار عملي: قبل أن تخلط بين الكفاءة الحسابية والصحة البيولوجية، تأكد من أن التحليل الإحصائي والتجريبي يدعمان استنتاجاتك.

مثال تطبيقي على خوارزمية محاذاة باستخدام البرمجة الديناميكية (Needleman-Wunsch):

· المدخلات: تسلسلان DNA (أو بروتين)، مصفوفة استبدال، تكاليف فجوات.

· الافتراضات: أن المحاذاة العالمية هي الأكثر ملاءمة، وأن تكاليف الفجوات تعكس الواقع البيولوجي.

· المخرجات: محاذاة مثالية مع درجة تشابه.

· القيود: التعقيد O(n*m) يجعلها غير عملية للتسلسلات الطويلة (أكثر من 10,000 قاعدة) دون تحسينات أو فهرسة.

Applied example of a dynamic programming alignment algorithm (Needleman-Wunsch):

اختبار: قبل اختيار خوارزمية، قم بتقدير حجم المدخلات وتأكد من أن التعقيد مقبول ضمن الموارد المتاحة. إذا لم يكن، فابحث عن خوارزميات بديلة أو تقنيات تحسين.

· Inputs: two DNA (or protein) sequences, a substitution matrix, gap costs.

· Assumptions: that global alignment is most appropriate, and that gap costs reflect biological reality.

· Outputs: an optimal alignment with a similarity score.

· Limitations: O(n*m) complexity makes it impractical for long sequences (over 10,000 bases) without optimizations or indexing.

Test: before choosing an algorithm, estimate the input size and ensure that the complexity is acceptable within available resources. If not, look for alternative algorithms or optimization techniques.

عند اختيار خوارزمية، يجب تسجيل ما يلي:

· المصدر: الخوارزمية المستخدمة (مرجعها النظري أو تنفيذها العملي).

· الإصدار: إصدار المكتبة أو الأداة المستخدمة.

· المعلمات: أي معاملات تؤثر على التعقيد (مثل حجم الفهرسة، أو حجم الدفعة).

· التحويلات: أي تعديلات على المدخلات لتقليل التعقيد (مثل التصفية المسبقة، أو استخدام عينة فرعية).

When choosing an algorithm, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع فهم سبب اختيار هذه الخوارزمية وما هي حدودها، ويعيد تطبيقها بنفس المعلمات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the algorithm used (its theoretical reference or practical implementation).

· Version: the version of the library or tool used.

· Parameters: any parameters affecting complexity (e.g., index size, batch size).

· Transformations: any modifications to inputs to reduce complexity (e.g., pre-filtering, or using a subsample).

Test: Practical check: "If I gave this record to a colleague a year later, could they understand why this algorithm was chosen and what its limits are, and reapply it with the same parameters?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على خوارزمية في تحليل بيانات حقيقية، اختبرها على بيانات اصطناعية صغيرة ومتوسطة الحجم. اختبر الخوارزمية على مدخلات ذات أحجام مختلفة، وقم بقياس الزمن والذاكرة. تحقق من أن النمو يتوافق مع التعقيد النظري المتوقع. إذا كان النمو أسرع من المتوقع، فقد يكون هناك خطأ في التنفيذ أو اختيار غير مناسب للخوارزمية.

هذا الاختبار يمنحك الثقة أن الخوارزمية ستعمل بشكل مقبول على البيانات الحقيقية.

Before relying on an algorithm in real data analysis, test it on small and medium-sized synthetic data. Test the algorithm on inputs of different sizes, and measure time and memory. Verify that the growth matches the expected theoretical complexity. If the growth is faster than expected, there may be an implementation error or inappropriate algorithm choice. This test gives you confidence that the algorithm will perform acceptably on real data.

Part 8 — CRISPR from a Scientific and Secure Computational Perspective

الباب 8 — CRISPR من منظور علمي وحاسوبي آمن

Chapter 27 — Clustering

الفصل

الفكرة المركزية: المسافة، التجميع، اختيار الخوارزمية، الاستقرار والتفسير.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Clustering discovers structure under a chosen distance metric, algorithm, and set of assumptions. There is no universally correct clustering; different algorithms can produce different partitions while remaining computationally valid. Stability, outliers, distance choice, and metadata relationships therefore need inspection.

التجميع يكتشف بنية في البيانات وفق مقياس مسافة وخوارزمية وافتراضات محددة. لا توجد عناقيد صحيحة بصورة مطلقة؛ يمكن لخوارزميات مختلفة أن تعطي تقسيمات مختلفة، وكلها قد تكون صحيحة حسابياً. لذلك يجب فحص الاستقرار، والقيم الشاذة، واختيار المسافة، والعلاقة مع البيانات الوصفية.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. What Is Clustering؟

1. ما هو التجميع (Clustering)؟

Clustering is an unsupervised machine learning technique that aims to partition a set of data into groups (clusters) such that elements within each cluster are similar, and elements between different clusters are dissimilar. In bioinformatics, clustering is used in many contexts: grouping genes with similar expression patterns, grouping samples by clinical characteristics, grouping sequences by similarity, and identifying subtypes in genomic data.

التجميع هو أسلوب تعلم آلي غير موجّه (unsupervised learning) يهدف إلى تقسيم مجموعة من البيانات إلى مجموعات (clusters) بحيث تكون العناصر داخل كل مجموعة متشابهة، والعناصر بين المجموعات المختلفة مختلفة. في المعلوماتية الحيوية، يُستخدم التجميع في العديد من السياقات: تجميع الجينات ذات أنماط التعبير المتشابهة، تجميع العينات حسب الخصائص السريرية، تجميع التسلسلات حسب التشابه، وتحديد الأنواع الفرعية في بيانات الجينوم.

مثال بسيط:

لديك بيانات عن 5 عينات، كل منها لها قياسان (X, Y).

العينات: (2,3), (2.5,2.8), (3,3.2), (8,9), (8.5,8.8)

التجميع قد يضع المجموعة الأولى (الثلاث عينات الأولى) في مجموعة، والمجموعة الثانية (العينتان الأخيرتان) في مجموعة أخرى.

2. Fundamental Concepts in Clustering

2. مفاهيم أساسية في التجميع

مقياس المسافة (Distance Measure) : يحدد كيف نقيس التشابه بين عينتين. اختيار المقياس يؤثر على شكل المجموعات. مقاييس شائعة في المعلوماتية الحيوية تشمل المسافة الإقليدية (Euclidean)، مسافة مانهاتن (Manhattan)، مسافة الارتباط (Correlation distance)، ومسافة Jaccard للبيانات الثنائية.

التطبيع (Normalization) : قبل التجميع، غالباً ما تُطبّق تطبيع على البيانات لجعلها قابلة للمقارنة. في بيانات التعبير الجيني، قد تُطبّق تحويل لوغاريتمي وتطبيع بواسطة Z-score لجعل توزيع كل جين متوسطه صفر وانحرافه المعياري واحد. التطبيع يؤثر على المسافات بين العينات وبالتالي على المجموعات الناتجة.

خوارزمية التجميع (Clustering Algorithm) : الطريقة التي تُستخدم لتشكيل المجموعات. نوعان رئيسيان:

Distance Measure: defines how we measure similarity between two samples. The choice of measure affects the shape of clusters. Common measures in bioinformatics include Euclidean distance, Manhattan distance, Correlation distance, and Jaccard distance for binary data.

· التجميع الهرمي (Hierarchical Clustering): يبني شجرة (dendrogram) من المجموعات المتداخلة. لا يتطلب تحديد عدد المجموعات مسبقاً، لكنك تقطع الشجرة عند مستوى معين للحصول على مجموعات.

Normalization: before clustering, data is often normalized to make it comparable. In gene expression data, log transformation and Z-score normalization (mean 0, standard deviation 1) may be applied. Normalization affects the distances between samples and therefore the resulting clusters.

· التجميع التقسيمي (Partitional Clustering): يقسم البيانات إلى عدد محدد مسبقاً من المجموعات. أشهر مثال هو K-means.

Clustering Algorithm: the method used to form clusters. Two main types:

· Hierarchical Clustering: builds a tree (dendrogram) of nested clusters. It does not require specifying the number of clusters in advance, but you cut the tree at a certain level to obtain clusters.

· Partitional Clustering: partitions the data into a pre-specified number of clusters. The most famous example is K-means.

3. Clustering as an Exploratory Tool

3. التجميع كأداة استكشافية

التجميع هو أداة استكشافية، وليس أداة اختبار فرضيات. لا يُستخدم التجميع لإثبات وجود مجموعات بيولوجية؛ بل يُستخدم لتوليد فرضيات حول بنية البيانات التي يمكن اختبارها لاحقاً بطرق مستقلة. لذلك، يجب أن تكون حذراً في تفسير نتائج التجميع. ظهور مجموعتين لا يعني وجود مجموعتين بيولوجيتين حقيقيتين. قد يكون السبب:

· اختيار مقياس مسافة غير مناسب

· تطبيع غير صحيح

· وجود قيم شاذة (outliers) تؤثر على التجميع.

· اختيار عدد غير مناسب من المجموعات.

Clustering is an exploratory tool, not a hypothesis-testing tool. Clustering is not used to prove the existence of biological groups; it is used to generate hypotheses about data structure that can later be tested by independent methods. Therefore, it is appropriate to be cautious in interpreting clustering results. The appearance of two clusters does not mean there are two real biological groups. The cause may be:

· ضوضاء تقنية (batch effects)

· Inappropriate distance measure.

· Incorrect normalization

· Presence of outliers affecting clustering.

· Inappropriate number of clusters.

· Technical noise (batch effects).

4. Choosing the Number of Clusters

4. اختيار عدد المجموعات

في التجميع التقسيمي، يجب تحديد عدد المجموعات مسبقاً. هذا قرار صعب، وليس له إجابة واحدة صحيحة. هناك عدة طرق لتقدير العدد المناسب:

· طريقة الكوع (Elbow Method): في K-means، ترسم مجموع المربعات داخل المجموعات (Within-Cluster Sum of Squares) مقابل عدد المجموعات. تبحث عن نقطة "الكوع" حيث يتوقف الانخفاض الحاد.

In partitional clustering, you must specify the number of clusters in advance. This is a difficult decision with no single correct answer. Several methods can estimate the appropriate number:

· معامل الصورة الظلية (Silhouette Coefficient): يقيس مدى تشابه كل نقطة مع مجموعتها مقارنة بالمجموعات الأخرى. يتراوح بين -1 و 1، والقيم الأعلى تشير إلى تجميع أفضل.

· Elbow Method: In K-means, plot the Within-Cluster Sum of Squares against the number of clusters. Look for the "elbow" point where the sharp decrease stops.

· المعرفة البيولوجية (Biological Knowledge): في بعض الأحيان، تعرف مسبقاً عدد المجموعات المتوقعة (مثل أنواع فرعية معروفة من السرطان).

· Silhouette Coefficient: measures how similar each point is to its own cluster compared to other clusters. Ranges from -1 to 1, with higher values indicating better clustering.

· Biological Knowledge: sometimes you know the expected number of clusters in advance (e.g., known cancer subtypes).

5. Cluster Stability

5. استقرار المجموعات (Cluster Stability)

أحد أهم الاختبارات التي يجب تطبيقها على أي نتيجة تجميع هو اختبار الاستقرار. إذا تغيرت المجموعات جذرياً عند تغيير مقياس المسافة، أو التطبيع، أو حذف عينة واحدة، فهذه إشارة إلى أن البنية المكتشفة غير قوية. اختبار الاستقرار يتم عادةً بواسطة:

· أخذ عينات متعددة (Subsampling): تكرار التجميع على عينات فرعية عشوائية من البيانات، وقياس مدى تشابه المجموعات الناتجة.

One of the most important tests to apply to any clustering result is stability testing. If clusters change drastically when you change the distance measure, normalization, or remove a single sample, it is a sign that the discovered structure is not robust. Stability testing is usually done by:

· تغيير المعلمات: تكرار التجميع باستخدام مقاييس مسافة مختلفة، أو خوارزميات مختلفة، ومراقبة التغيرات.

· Subsampling: repeating clustering on random subsamples of the data, and measuring the similarity of the resulting clusters.

· تحليل الحساسية (Sensitivity Analysis): إزالة عينة واحدة في كل مرة وتكرار التجميع، لمعرفة أي العينات تؤثر بشدة على النتيجة.

· Changing parameters: repeating clustering with different distance measures or different algorithms, and observing changes.

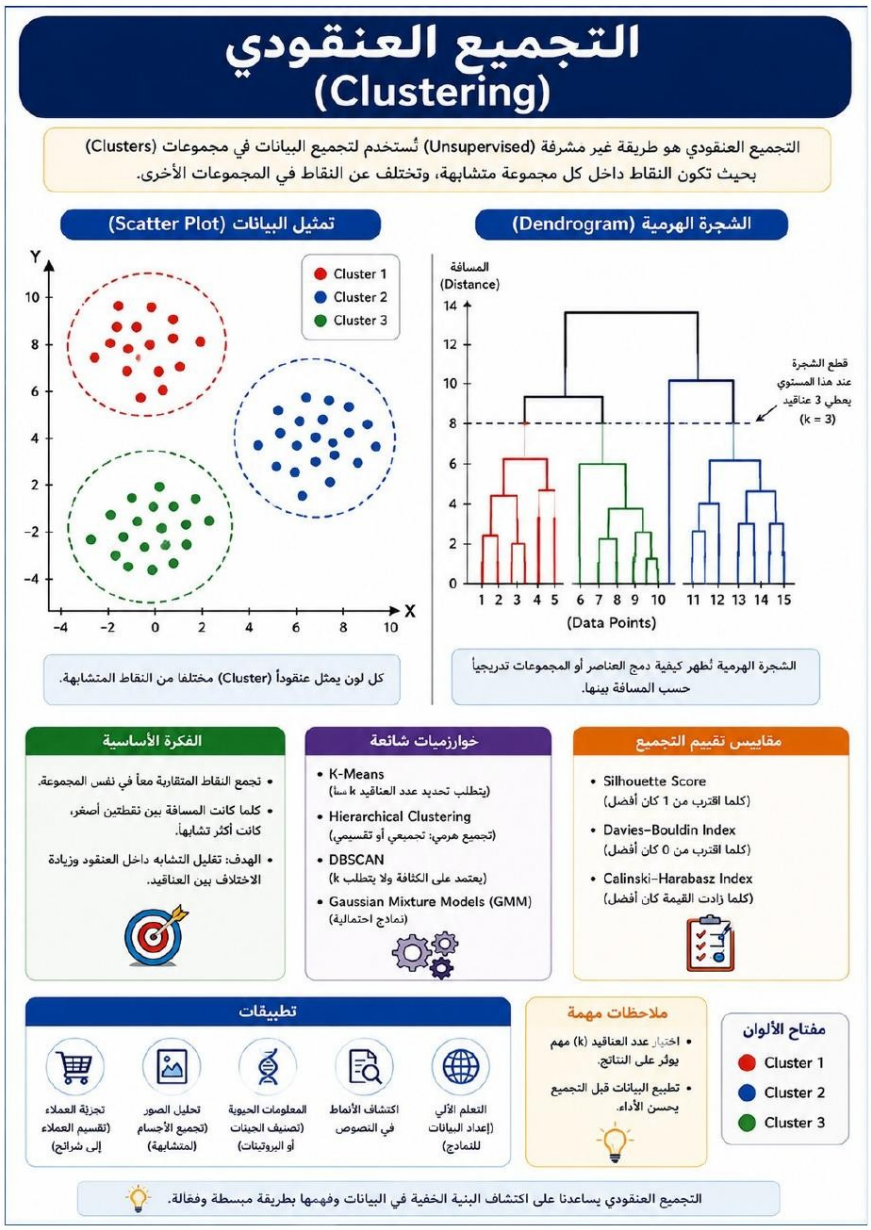
· Sensitivity Analysis: removing one sample at a time and repeating clustering, to see which samples strongly affect the result.

6. Practical Application: Clustering Gene Expression Data

Let's apply clustering to simple gene expression data. We have 4 genes and 6 samples (3 healthy and 3 diseased). We want to see if samples cluster by condition.

6. التطبيق العملي: تجميع بيانات تعبير جيني

لنطبق التجميع على بيانات تعبير جيني بسيطة. لدينا 4 جينات و 6 عينات (3 سليمة و 3 مريضة). نريد أن نرى ما إذا كانت العينات تتجمع حسب الحالة.



إنفوجرافيك 22 / *Infographic 22*

Purpose: turn numerical results into a visual representation for inspecting structure and differences.

الغرض: تحويل النتائج الرقمية إلى تمثيل بصري يساعد على فحص البنية والاختلافات.

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
# بيانات تعبير افتراضية: 4 جينات x 6 عينات
# العينات 0-2 سليمة، 3-5 مريضة
data = np.array([
    [10, 9, 11, 15, 16, 14],  # جين 1
    [8, 7, 9, 12, 13, 11],    # جين 2
    [20, 19, 21, 22, 23, 20], # جين 3
    [5, 6, 5, 8, 9, 7]        # جين 4
])  # T. (6 عينات x 4 جينات)
# تطبيع البيانات
scaler = StandardScaler()
data_norm = scaler.fit_transform(data)
# تطبيق K-means (2 مجموعات)
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
clusters = kmeans.fit_predict(data_norm)
print("تسميات المجموعات:", clusters)
print("توزيع العينات حسب الحالة الفعلية:")
print("عينات سليمة (0-2):", clusters[:3])
print("عينات مريضة (3-5):", clusters[3:])
# تصور باستخدام PCA
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_norm)
plt.figure(figsize=(8, 6))
scatter = plt.scatter(data_pca[:, 0], data_pca[:, 1], c=clusters, cmap='viridis')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Clustering of Samples')
plt.colorbar(label='Cluster')
plt.show()
```

In this simple example, we assume clustering separates healthy and diseased samples, but this depends on the strength of the biological signal and the absence

of technical effects. If technical effects are strong, samples may cluster by sequencing batch instead of condition.

7. Limits of Clustering and Its Interpretation

في هذا المثال البسيط، نفترض أن التجميع يفصل بين العينات السليمة والمريضة، لكن هذا يعتمد على قوة الإشارة البيولوجية وغياب التأثيرات التقنية. إذا كانت التأثيرات التقنية قوية، قد تتجمع العينات حسب دفعة التسلسل بدلاً من الحالة.

7. حدود التجميع وتفسيره

التجميع أداة قوية، لكن لها حدود:

· لا يوجد معيار مطلق للصواب: لا توجد طريقة واحدة صحيحة للتجميع. خوارزميات مختلفة تعطي مجموعات مختلفة، وكلها قد تكون صحيحة حسابياً لكنها تروي قصصاً مختلفة.

· التجميع لا يعني السببية: وجود مجموعتين لا يعني أن هناك آلية بيولوجية تسبب هذا الانفصال.

Clustering is a powerful tool, but it has limits:

· التجميع حساس للقيم الشاذة: عينة واحدة شاذة قد تغير شكل المجموعات بشكل كبير.

· No absolute standard of correctness: there is no single correct way to cluster. Different algorithms give different clusters, all of which may be computationally correct but tell different stories.

· التجميع في الأبعاد العالية: كلما زاد عدد الأبعاد (الجينات)، تصبح المسافات أقل معنى (لعنة الأبعاد - curse of dimensionality).

· Clustering does not imply causation: the presence of two clusters does not mean there is a biological mechanism causing this separation.

· Clustering is sensitive to outliers: a single outlier sample may significantly change cluster shapes.

· Clustering in high dimensions: as the number of dimensions (genes) increases, distances become less meaningful (curse of dimensionality).

Chapter Summary

خلاصة الفصل

Clustering is an exploratory statistical technique for partitioning data into groups of similar elements. It depends on the choice of distance measure, normalization, clustering algorithm, and number of clusters. Clustering is a tool for generating hypotheses, not for proving them. The appearance of clusters in data does not

necessarily mean there are real biological groups; it may be a result of technical choices or noise. Testing cluster stability is essential before any biological interpretation. The next chapter will move to CRISPR, from a scientific and secure computational perspective.

التجميع هو أسلوب إحصائي استكشافي لتقسيم البيانات إلى مجموعات من العناصر المتشابهة. يعتمد على اختيار مقياس المسافة، والتطبيع، وخوارزمية التجميع، وعدد المجموعات. التجميع هو أداة لتوليد الفرضيات، وليس لإثباتها. ظهور مجموعات في البيانات لا يعني بالضرورة وجود مجموعات بيولوجية حقيقية؛ فقد يكون نتيجة لاختيارات تقنية أو ضوضاء. اختبار استقرار المجموعات أمر ضروري قبل أي تفسير بيولوجي. الفصل القادم سينتقل إلى CRISPR، من منظور علمي وحاسوبي آمن.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Clustering" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Clustering" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "the presence of two clusters in clustering analysis proves the existence of two biological subtypes." This is incorrect; clustering is an exploratory tool, and the result depends on technical choices. The correct statement is: "clustering identifies potential groups in the data, but proving the existence of real biological groups requires independent testing and stability verification." A simple test: Practical check: "Do the clusters I found remain stable when changing the distance measure or normalization?".

مثال على التجاوز: القول إن "وجود مجموعتين في تحليل التجميع يثبت وجود نوعين فرعيين بيولوجيين". هذا غير صحيح؛ التجميع أداة استكشافية، والنتيجة تعتمد على اختيارات تقنية. الصحيح هو: "التجميع يحدد مجموعات محتملة في البيانات، لكن إثبات وجود مجموعات بيولوجية حقيقية يتطلب اختبارات مستقلة وتحققاً من الاستقرار." اختبار بسيط: فحص تطبيقي: "هل المجموعات التي وجدتها تبقى ثابتة عند تغيير مقياس المسافة أو التطبيع؟".

Example: A K-means result giving two clusters is a computational representation. Claiming that "these two clusters represent two disease subtypes" is a biological claim not proven by clustering alone. Clustering places samples into groups based on distances, but it does not determine whether these groups have biological meaning. Practical test: before inferring a subtype from clustering, check cluster stability and look for independent evidence (such as clinical markers, or pathway analysis).

مثال: نتيجة تحليل K-means تعطي مجموعتين هو تمثيل حسابي. الادعاء بأن "هاتين المجموعتين تمثلان نوعين فرعيين من المرض" هو ادعاء بيولوجي لا يثبته التجميع وحده. التجميع يضع العينات في مجموعات بناءً على المسافات، لكنه لا يحدد ما إذا كانت هذه المجموعات ذات معنى بيولوجي. اختبار عملي: قبل أن تستنتج وجود نوع فرعي من التجميع، تحقق من استقرار المجموعات وابحث عن أدلة مستقلة (مثل علامات سريرية، أو تحليل مسارات).

مثال تطبيقي على K-means:

· المدخلات: بيانات رقمية (مصفوفة عينات × ميزات)، عدد المجموعات (k)، مقياس المسافة (افتراضياً إقليدي)، عدد التكرارات.

· الافتراضات: أن المجموعات كروية ومتساوية الحجم (في K-means الأساسي)، وأن التطبيع قد تم بشكل مناسب.

· المخرجات: تسميات المجموعات لكل عينة، ومراكز المجموعات.

· القيود: K-means حساس للقيم الشاذة، ويتطلب تحديد k مسبقاً، ويفترض مجموعات كروية. قد لا يعمل بشكل جيد مع مجموعات ذات أشكال معقدة أو أحجام مختلفة.

Applied example of K-means:

اختبار: قبل استخدام K-means، حاول تصور البيانات (باستخدام PCA أو t-SNE) لتقدير ما إذا كانت المجموعات كروية، وجرب قيماً مختلفة لـ k واستخدم معامل الصورة الظلية لتقييم الجودة.

· Inputs: numerical data (samples × features matrix), number of clusters (k), distance measure (default Euclidean), number of iterations.

· Assumptions: that clusters are spherical and equal-sized (in basic K-means), and that normalization has been properly performed.

· Outputs: cluster labels for each sample, and cluster centers.

· Limitations: K-means is sensitive to outliers, requires pre-specifying k, and assumes spherical clusters. It may not work well with complex-shaped or differently sized clusters.

Test: before using K-means, try visualizing the data (using PCA or t-SNE) to estimate whether clusters are spherical, and try different values of k using the silhouette coefficient to evaluate quality.

عند استخدام التجميع، يجب تسجيل ما يلي:

· المصدر: البيانات المستخدمة (مصدرها، تاريخ الحصول عليها، أي تصفية مسبقة).

· الإصدار: إصدار المكتبة المستخدمة (مثل scikit-learn، Bioconductor).

· المعلمات: خوارزمية التجميع، مقياس المسافة، عدد المجموعات، طريقة التطبيع، أي معاملات أخرى.

· التحويلات: أي خطوات تطبيع أو تحويل طبقت على البيانات قبل التجميع.

When using clustering, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة التجميع والحصول على نفس المجموعات دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the data used (its source, date obtained, any pre-filtering).

· Version: the library version used (e.g., scikit-learn, Bioconductor).

· Parameters: clustering algorithm, distance measure, number of clusters, normalization method, and any other parameters.

· Transformations: any normalization or transformation steps applied to the data before clustering.

Test: Practical check: "If I gave this record to a colleague a year later, could they re-run the clustering and obtain the same clusters without my help?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على التجميع في بيانات حقيقية، اختبر خوارزمية التجميع على بيانات اصطناعية ذات مجموعات معروفة. أنشئ بيانات تحتوي على مجموعتين أو ثلاث مجموعات واضحة، مع إضافة بعض الضوضاء. شغّل التجميع وتحقق من أنه يكتشف المجموعات المعروفة بدقة (حساسية ونوعية). إذا نجح الاختبار، فأنت واثق من أن الخوارزمية تعمل كما هو متوقع مع البيانات ذات البنية الواضحة. ثم جرب تغيير المعلمات (مثل عدد المجموعات، مقياس المسافة) وشاهد كيف تتغير النتائج. هذا يعلمك حساسية الخوارزمية قبل أن تواجه بيانات حقيقية معقدة.

Before relying on clustering on real data, test the clustering algorithm on synthetic data with known clusters. Create data containing two or three clear clusters, with some noise added. Run clustering and verify that it accurately detects the known clusters (sensitivity and specificity). If the test succeeds, the workflow is confident that the algorithm works as expected on data with clear structure. Then try changing parameters (such as number of clusters, distance measure) and see how results change. This teaches you the algorithm's sensitivity before you face complex real data.

الفكرة المركزية: المناعة التكيفية، guide RNA، Cas9، التاريخ العلمي، الاستخدام المسؤول.

Chapter 28 — CRISPR-Cas9 History and Conceptual Mechanism

الفصل 28 — تاريخ CRISPR-Cas9 ومبدأه

شرح أعمق للمفهوم

Deeper Conceptual Explanation

CRISPR-Cas9 مثال ممتاز على الفرق بين فهم الآلية وبناء نظام تطبيقي. يمكن شرح فكرة التعرف على تسلسل الهدف والقطع بصورة مفاهيمية، لكن الانتقال إلى تصميم عملي يتطلب سياقاً تجريبياً وضوابط وتقييماً للمخاطر. في هذا الكتاب يبقى التركيز على التمثيل الحسابي والمبادئ الوقائية.

CRISPR-Cas9 clearly illustrates the difference between understanding a mechanism and building an applied system. The conceptual idea of target recognition and cutting can be explained, but practical design requires experimental context, controls, and risk assessment. The book therefore keeps its emphasis on computational representation and safety principles.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. The Beginning: Mysterious Sequences in E. coli

1. البداية: تسلسلات غامضة في بكتيريا E. coli

The CRISPR story begins in 1987, when a Japanese research team led by Yoshizumi Ishino was analyzing a gene responsible for alkaline phosphatase isozyme conversion in Escherichia coli bacteria. During this analysis, the team noticed unusual repeated sequences in the bacterial DNA. These sequences repeated regularly, but no one had any idea about their function. At that time, sequencing tools were not as advanced as they are today, and sequencing these difficult-to-study DNA fragments took several months.

قصة CRISPR تبدأ في عام 1987، عندما كان فريق بحث ياباني بقيادة يوشيزومي إيشينو (Yoshizumi Ishino) يحلل جيناً مسؤولاً عن تحويل إنزيم الفوسفاتاز القلوي في بكتيريا الإشريكية القولونية (E. coli). خلال هذا التحليل، لاحظ الفريق وجود تسلسلات متكررة غير عادية في الحمض النووي للبكتيريا. كانت هذه التسلسلات تتكرر بشكل منتظم، لكن لم يكن لأحد أي فكرة عن وظيفتها. في ذلك الوقت، لم تكن أدوات التسلسل متطورة كما هي اليوم، واستغرق تسلسل هذه الأجزاء الصعبة من الحمض النووي عدة أشهر.

In 1993, Francisco Mojica from the University of Alicante in Spain was the first researcher to characterize these repeated sequences in another organism, finding

similar structures in the archaeal genome of Haloferax mediterranei. The presence of these sequences in two evolutionarily remote domains of life (bacteria and archaea) suggested their great functional significance, and served as an impetus for further research. In 2000, Mojica recognized that what had been reported as disparate repeat sequences actually shared a common set of features, now known to be hallmarks of CRISPR sequences. He also coined the term "CRISPR" through correspondence with Ruud Jansen, who first used the term in print in 2002.

في عام 1993، كان فرانسيسكو موخيكا (Francisco Mojica) من جامعة أليكانتي في إسبانيا أول باحث يحدد هذه التسلسلات المتكررة في كائن حي آخر، حيث عثر على بنى مشابهة في الجينوم العتائقي لـ Haloferax mediterranei. وجود هذه التسلسلات في مجالين تطوريين بعيدين (البكتيريا والعتائق) أشار إلى أهميتها الوظيفية الكبيرة، ودفع الباحثين لمواصلة الدراسة. في عام 2000، أدرك موخيكا أن ما تم الإبلاغ عنه كتسلسلات متكررة متنافرة تشترك في مجموعة مشتركة من السمات، التي تُعرف الآن بسمات تسلسلات CRISPR. كما صاغ مصطلح "CRISPR" من خلال مراسلات مع رود يانسن (Ruud Jansen)، الذي استخدم المصطلح لأول مرة في منشور علمي عام 2002.

2. Discovery of Function: An Adaptive Immune System

2. اكتشاف الوظيفة: نظام مناعي تكيفي

The real breakthrough came in 2005, when Mojica and his colleagues discovered similarity between the spacer regions of CRISPRs and sequences from bacteriophages, archaeal viruses, and plasmids. In other words, bacteria were storing pieces of viral DNA in their own genome, like an "ID card" for viruses they had encountered. This led Mojica to the revolutionary hypothesis: that the CRISPR system is an adaptive immune system in bacteria and archaea, protecting them from viruses and mobile genetic elements.

الاختراق الحقيقي جاء في عام 2005، عندما اكتشف موخيكا وزملاؤه تشابهاً بين الفواصل (spacers) في مناطق CRISPR وتسلسلات من جينومات العاثيات (bacteriophages) والفيروسات العتائقية والبلازميدات. بعبارة أخرى، البكتيريا كانت تخزن قطعاً من الحمض النووي الفيروسي في جينومها الخاص، مثل "بطاقة هوية" للفيروسات التي واجهتها. هذا قاد موخيكا إلى الفرضية الثورية: أن نظام CRISPR هو نظام مناعي تكيفي في البكتيريا والعتائق، يحميها من الفيروسات والعناصر الوراثية المتحركة.

Around the same time, Alexander Bolotin at the French National Institute for Agricultural Research discovered a new protein in Streptococcus thermophilus bacteria, a large protein he predicted to have nuclease activity. This protein is now known as Cas9. Bolotin also noted a short common sequence at the end of all

spacers, now known as the PAM (Protospacer Adjacent Motif), which is required for target recognition.

في نفس العام تقريباً، اكتشف ألكسندر بولوتين (Alexander Bolotin) في المعهد الفرنسي للبحوث الزراعية وجود بروتين جديد في بكتيريا Streptococcus thermophilus، وهو بروتين كبير تنبأ بأنه يمتلك نشاط نوكلياز. هذا البروتين يُعرف الآن باسم Cas9. كما لاحظ بولوتين وجود تسلسل قصير مشترك في نهاية جميع الفواصل، وهو ما يُعرف الآن بـ PAM (Protospacer Adjacent Motif)، وهو تسلسل ضروري للتعرف على الهدف.

In 2007, Philippe Horvath and his colleagues at Danisco experimentally proved that the CRISPR system is indeed an adaptive immune system: bacteria integrate new viral DNA into the CRISPR array, which allows them to fight off the next wave of attacking phage. They also showed that Cas9 is likely the only protein required for interference, the process by which the CRISPR system inactivates invading phage.

في عام 2007، أثبت فيليب هورفاث (Philippe Horvath) وزملاؤه في شركة Danisco تجريبياً أن نظام CRISPR هو بالفعل نظام مناعة تكيفية: البكتيريا تدمج حمضاً نووياً فيروسياً جديداً في مصفوفة CRISPR، مما يسمح لها بمحاربة الموجة التالية من الفيروسات المهاجمة. وأظهروا أيضاً أن بروتين Cas9 هو على الأرجح البروتين الوحيد المطلوب لتعطيل الفيروسات الغازية.

3. From Immune System to Gene-Editing Tool

3. من نظام مناعي إلى أداة تعديل جيني

The crucial step toward transforming CRISPR into a gene-editing tool came in 2012, when Emmanuelle Charpentier, Jennifer Doudna, and their team published a paper in Science. The team showed that the CRISPR-Cas9 system relies on two RNA molecules: crRNA (which carries the target sequence) and tracrRNA (which helps guide Cas9). Most importantly, they proved that these two molecules could be fused into a single molecule called a "single guide RNA" (sgRNA), greatly simplifying the system. Thus, it became possible to program the Cas9 nuclease to cut any desired DNA sequence.

الخطوة الحاسمة نحو تحويل CRISPR إلى أداة تعديل جيني جاءت في عام 2012، عندما نشرت إيمانويل شاربنتييه (Emmanuelle Charpentier) وجينيفر دودنا (Jennifer Doudna) وفريقهما ورقة بحثية في مجلة Science. أظهر الفريق أن نظام CRISPR-Cas9 يعتمد على جزيئين من RNA: crRNA (الذي يحمل التسلسل المستهدف) و tracrRNA (الذي يساعد في توجيه Cas9). الأهم من ذلك، أثبتوا أنه يمكن دمج هذين الجزيئين في

جزيء واحد يسمى "الدليل المفرد" (sgRNA أو single guide RNA)، مما يبسط النظام بشكل كبير. وبهذا، أصبح بالإمكان برمجة نوكلياز Cas9 لقطع أي تسلسل DNA مرغوب.

In 2013, Feng Zhang at the Broad Institute was the first to successfully apply the CRISPR-Cas9 system in eukaryotic cells (human and mouse cells). This opened the door to using CRISPR in medicine, agriculture, and basic research. In 2020, the Nobel Prize in Chemistry was awarded to Charpentier and Doudna in recognition of their development of "a method for genome editing".

في عام 2013، كان فينغ تشانغ (Feng Zhang) من معهد برود أول من طبق نظام CRISPR-Cas9 بنجاح في الخلايا حقيقية النوى (الخلايا البشرية والفأرية). هذا فتح الباب أمام استخدام CRISPR في الطب والزراعة والبحث الأساسي. في عام 2020، مُنحت جائزة نوبل في الكيمياء لشاربنتييه ودودنا تقديراً لتطويرهما "طريقة لتحرير الجينوم".

4. The General Principle: How Does CRISPR-Cas9 Work؟

4. المبدأ العام: كيف يعمل CRISPR-Cas9؟

To understand how CRISPR-Cas9 works, we can divide the process into simple steps:

لفهم كيف يعمل CRISPR-Cas9، يمكننا تقسيم العملية إلى خطوات بسيطة:

Step One: Designing the Guide (Guide RNA).

الخطوة الأولى: تصميم الدليل (Guide RNA).

A short RNA molecule (about 20 nucleotides) is designed to match the target sequence in the genome. This is the "guide" that will lead Cas9 to the correct site.

يتم تصميم جزيء RNA قصير (حوالي 20 نيوكليوتيدة) يتطابق مع التسلسل المستهدف في الجينوم. هذا هو "الدليل" الذي سيقود Cas9 إلى الموقع الصحيح.

Step Two: PAM Recognition

الخطوة الثانية: التعرف على PAM

For Cas9 to work, there must be a short sequence adjacent to the target site called PAM (Protospacer Adjacent Motif). This sequence (usually NGG) acts like a "security key": without it, Cas9 cannot bind to the DNA. The presence of PAM ensures that Cas9 does not accidentally cut the bacterium's own DNA.

لكي يعمل Cas9، يجب أن يكون هناك تسلسل قصير بجوار الموقع المستهدف يسمى PAM (Protospacer Adjacent Motif). هذا التسلسل (عادة NGG) يعمل مثل "مفتاح أمان": بدونه، لا يستطيع Cas9 الارتباط بالـ DNA. وجود PAM يضمن أن Cas9 لا يقطع الحمض النووي للبكتيريا نفسها عن طريق الخطأ.

Step Three: Binding and Cleavage.

الخطوة الثالثة: الارتباط والقطع.

The Cas9-guide RNA complex binds to the DNA at the target site. The double-stranded DNA temporarily unwinds, and the guide RNA forms base pairs with the complementary strand. Then, Cas9 cuts both strands at a specific location (three bases upstream of PAM), creating a double-strand break in the DNA.

يرتبط مركب Cas9-guide RNA بالـ DNA في الموقع المستهدف. يتفكك الـ DNA مزدوج الشريطة مؤقتاً، ويشكل الدليل RNA أزواجاً قاعدية مع الشريط المكمل. بعد ذلك، يقوم Cas9 بقطع كلا الشريطين في موقع محدد (ثلاث قواعد قبل PAM)، محدثاً كسراً مزدوجاً في الـ DNA (Double-Strand Break).

الخطوة الرابعة: إصلاح الخلية

الخلية لديها آليات لإصلاح الكسور المزدوجة في الـ DNA. هناك مساران رئيسيان:

Step Four: Cell Repair

· الانضمام الطرفي غير المماثل (NHEJ): إصلاح سريع لكنه غير دقيق، يؤدي غالباً إلى حذف أو إضافة قواعد، مما يعطل الجين (Gene Knockout).

The cell has mechanisms to repair double-strand breaks in DNA. There are two main pathways:

· الإصلاح الموجه بالتماثل (HDR): إصلاح دقيق باستخدام قالب، يسمح بإدخال تعديل محدد في الجينوم.

· Non-Homologous End Joining (NHEJ): Fast but imprecise repair, often leading to deletion or addition of bases, resulting in gene disruption (Gene Knockout).

· Homology Directed Repair (HDR): Precise repair using a template, allowing the introduction of a specific modification into the genome.

5. Limits of Interpretation: Computational Representation Is Not Biological Proof

5. حدود التفسير: التمثيل الحسابي ليس دليلاً بيولوجياً

As with all bioinformatics tools, we must distinguish between computational representation and biological proof. the workflow can design guide RNAs using algorithms, predict cut sites, and even simulate the effects of modifications. But these simulations do not prove that the modification will occur in the living cell, or that it is safe, or that it leads to the desired outcome. Proving this requires laboratory experiments, analysis of off-target effects, and safety and ethical evaluation.

كما هو الحال مع كل أدوات المعلوماتية الحيوية، يجب أن نفرق بين التمثيل الحسابي والإثبات البيولوجي. يمكن تصميم guide RNA باستخدام خوارزميات، والتنبؤ بمواقع القطع، وحتى محاكاة تأثير التعديلات. لكن هذه المحاكاة لا تثبت أن التعديل سيحدث في الخلية الحية، أو أنه آمن، أو أنه يؤدي إلى النتيجة المرجوة. إثبات ذلك يتطلب تجارب معملية، وتحليلاً للنتائج غير المستهدفة (off-target effects)، وتقييماً للسلامة والأخلاقيات.

Chapter Summary

خلاصة الفصل

CRISPR-Cas9 is a remarkable story of transition from basic scientific discovery to a revolutionary technological tool. It began with mysterious sequences in E. coli bacteria in 1987, evolved through decades of research to understand its function as an adaptive immune system in bacteria, and culminated in its development as a programmable genome-editing tool in 2012. Its mechanism is simple: a guide RNA directs the Cas9 protein to a specific location in the genome, where it creates a double-strand break that the cell repairs in ways that either disrupt the gene or introduce a specific modification. But the immense power of this technology comes with responsibility: distinguishing between computational simulation and biological reality, and understanding the ethical and practical limits of its use. The next chapter will build on this foundation to explore the safe representation of locations using SeqFeature.

CRISPR-Cas9 هي قصة انتقال مذهل من اكتشاف علمي أساسي إلى أداة تقنية ثورية. بدأت بتسلسلات غامضة في بكتيريا E. coli عام 1987، وتطورت عبر عقود من البحث لفهم وظيفتها كنظام مناعي تكيفي في البكتيريا، وصولاً إلى تطويرها كأداة برمجية لتعديل الجينوم في عام 2012. مبدأ عملها بسيط: guide RNA يوجه بروتين Cas9 إلى موقع محدد في الجينوم، حيث يحدث قطعاً مزدوجاً يُصلحه الخلية بطرق تؤدي إما إلى تعطيل الجين أو إدخال تعديل محدد. لكن القوة الهائلة لهذه التقنية تأتي مع مسؤولية: الفصل بين المحاكاة الحاسوبية والحقيقة البيولوجية، وفهم الحدود الأخلاقية والعملية لاستخدامها. الفصل القادم سيبني على هذا الأساس لاستكشاف التمثيل الآمن للمواقع باستخدام SeqFeature.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "CRISPR-Cas9 History and Conceptual Mechanism" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "تاريخ CRISPR-Cas9 ومبدأه" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "CRISPR can edit any gene in any organism without errors." This is incorrect; CRISPR is effective but not perfect, and

may cut at off-target sites. The correct statement is: "CRISPR-Cas9 is a programmable system for cutting DNA at specific sites, but its accuracy depends on guide design, and there is a possibility of off-target effects that require careful examination." A simple test: Practical check: "Does guide RNA design guarantee that there is no other similar site in the genome that Cas9 might cut?".

مثال على التجاوز: القول إن "CRISPR يمكنه تعديل أي جين في أي كائن دون أخطاء". هذا غير صحيح؛ CRISPR فعال لكنه ليس مثالياً، وقد يحدث قطع في مواقع غير مستهدفة (off-target effects). الصحيح هو: "CRISPR-Cas9 هو نظام قابل للبرمجة لقطع DNA في مواقع محددة، لكن دقته تعتمد على تصميم الدليل، وهناك احتمالية لحدوث تأثيرات غير مستهدفة تتطلب فحصاً دقيقاً." اختبار بسيط: فحص تطبيقي: "هل تصميم guide RNA يضمن عدم وجود أي موقع آخر مشابه في الجينوم يمكن أن يقطعه Cas9؟"ـ

Example: Using software to design a guide RNA and predict the cut site is a computational representation. Claiming that "this guide will successfully edit the gene in the cell" is a biological claim not proven by the computational design alone. The design tells us the potential site, but the success of editing depends on delivery efficiency, cellular repair efficiency, and the absence of off-target effects. Practical test: before inferring successful editing from a computational design, verify that there is experimental evidence (such as in vitro or cell-based experiments) supporting this claim.

مثال: استخدام أداة برمجية لتصميم guide RNA والتنبؤ بموقع القطع هو تمثيل حسابي. الادعاء بأن "هذا الدليل سيعدل الجين بنجاح في الخلية" هو ادعاء بيولوجي لا يثبته التصميم الحسابي وحده. التصميم يخبرنا بالموقع المحتمل، لكن نجاح التعديل يعتمد على كفاءة التوصيل، وكفاءة الإصلاح الخلوي، وغياب التأثيرات غير المستهدفة. اختبار عملي: قبل أن تستنتج نجاح تعديل من تصميم حسابي، تحقق من وجود أدلة تجريبية (مثل تجارب في المختبر أو في الخلايا) تدعم هذا الادعاءـ

مثال تطبيقي على تصميم guide RNA:

· المدخلات: تسلسل DNA مستهدف (الجين المراد تعديله)، جينوم مرجعي للكائن الحي، تسلسل PAM المطلوبـ

· الافتراضات: أن التسلسل المستهدف فريد في الجينوم، وأن موقع PAM متاح، وأن guide RNA سيتصل بالخلية بكفاءةـ

· المخرجات: تسلسل guide RNA الموصى به، ومواقع القطع المتوقعة، وقائمة بالمواقع غير المستهدفة المحتملةـ

· القيود: التصميم الحسابي لا يضمن كفاءة القطع في الخلية، ولا يتنبأ بدقة بالتأثيرات غير المستهدفة في السياق الخلوي الحقيقي، ولا يأخذ في الاعتبار عوامل مثل تغليف الكروماتين أو تعديلات الـ DNAـ

Applied example of guide RNA design:

اختبار: قبل اعتماد تصميم guide RNA، تحقق من فريدة التسلسل المستهدف في الجينوم، وقيم قائمة المواقع غير المستهدفة المحتملة، وابحث عن أدلة تجريبية سابقة لاستخدام أدلة مماثلة.

· Inputs: target DNA sequence (the gene to be edited), a reference genome for the organism, the required PAM sequence.

· Assumptions: that the target sequence is unique in the genome, that the PAM site is available, and that the guide RNA will be delivered to the cell efficiently.

· Outputs: the recommended guide RNA sequence, predicted cut sites, and a list of potential off-target sites.

· Limitations: the computational design does not guarantee cutting efficiency in the cell, nor does it accurately predict off-target effects in the real cellular context, nor does it account for factors such as chromatin packaging or DNA modifications.

Test: before relying on a guide RNA design, verify the uniqueness of the target sequence in the genome, evaluate the list of potential off-target sites, and look for previous experimental evidence using similar guides.

عند استخدام أدوات تصميم guide RNA، يجب تسجيل ما يلي:

· المصدر: التسلسل المستهدف (رقم الانضمام، الكائن الحي، الجين)، والجينوم المرجعي المستخدم (الإصدار).

· الإصدار: إصدار الأداة المستخدمة (مثل CRISPOR، Benchling، CRISPRscan)، وإصدار الجينوم المرجعي.

· المعلمات: تسلسل PAM المستخدم، طول guide RNA، أي معاملات للكشف عن المواقع غير المستهدفة (مثل عدد الاختلافات المسموح بها).

· التحويلات: أي تعديلات على التسلسل المستهدف قبل التصميم (مثل تحويل إلى أحرف كبيرة، أو استبعاد مناطق معينة).

When using guide RNA design tools, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة تصميم نفس الـ guide RNA والحصول على نفس النتائج دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the target sequence (accession number, organism, gene), and the reference genome used (version).

· Version: the tool version used (e.g., CRISPOR, Benchling, CRISPRscan), and the reference genome version.

· Parameters: the PAM sequence used, guide RNA length, and any parameters for off-target detection (e.g., number of allowed mismatches).

· Transformations: any modifications to the target sequence before design (e.g., converting to uppercase, excluding certain regions).

Test: Practical check: "If I gave this record to a colleague a year later, could they redesign the same guide RNA and obtain the same results without my help?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على أداة تصميم guide RNA في مشروع حقيقي، اختبر الأداة على تسلسلات اصطناعية ذات مواقع معروفة. أنشئ تسلسلاً قصيراً يحتوي على موقع PAM وهدف واضح، وشغّل أداة التصميم. تحقق من أن الأداة تكتشف الموقع المستهدف وتنتج guide RNA مناسباً. ثم اختبر الأداة على تسلسل يحتوي على مواقع متعددة مشابهة، وتأكد من أنها تكتشف المواقع غير المستهدفة المحتملة. إذا نجحت هذه الاختبارات، فأنت واثق من أن الأداة تعمل كما هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس خصائص الجينوم وليس خطأ في الأداة.

Before relying on a guide RNA design tool in a real project, test the tool on synthetic sequences with known sites. Create a short sequence containing a PAM site and a clear target, and run the design tool. Verify that the tool detects the target site and produces an appropriate guide RNA. Then test the tool on a sequence containing multiple similar sites, and verify that it detects potential off-target sites. If these tests succeed, the workflow is confident that the tool works as expected, and any result with real data reflects the genome's properties, not an error in the tool.

Chapter 29 — Representing Features with SeqFeature

الفصل 29 — تمثيل المواقع باستخدام SeqFeature

الفكرة المركزية: الموقع كبيانات، الإحداثيات، الاتجاه، annotations، حدود التمثيل.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

عندما يمثل SeqFeature موقعاً مستهدفاً، فإنه يصف ما أُدخل إلى النموذج. لا يثبت ذيمكن أن الموقع صحيح في جينوم حقيقي أو أن الوظيفة المنسوبة إليه صحيحة. هذه الحدود مهمة لأن التمثيل قد يصبح لاحقاً مدخلاً لخوارزمية أخرى، وعندها تنتقل الأخطاء معه.

When SeqFeature represents a target location, it describes what was entered into the model. It does not prove that the location is correct in a real genome or that an

assigned function is valid. This boundary matters because an annotation may later feed another algorithm, carrying its errors forward.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. ما هو SeqFeature؟

1. What Is SeqFeature؟

SeqFeature is an object in Biopython that represents a specific region on a sequence with information about its location, strand, type, and additional properties. It is Biopython's way of translating "this part of the sequence is important for this reason" into a language that computers understand. In the context of CRISPR, SeqFeature can be used to represent the target site, the PAM site, the cut region, or any other feature on the genome. But the representation remains abstract: it is a description of the location, not proof that the location performs a specific function in the cell.

SeqFeature هو كائن في Biopython يمثل منطقة محددة على التسلسل مع معلومات عن موقعها واتجاهها ونوعها وخصائصها الإضافية. هو طريقة Biopython لترجمة "هذا الجزء من التسلسل مهم لهذا السبب" إلى لغة يفهمها الحاسوب. في سياق CRISPR، يمكن استخدام SeqFeature لتمثيل موقع الهدف، أو موقع PAM، أو منطقة القطع، أو أي ميزة أخرى على الجينوم. لكن التمثيل يبقى تجريدياً: هو وصف للموقع، وليس دليلاً على أن الموقع يؤدي وظيفة معينة في الخلية.

from Bio.SeqFeature import SeqFeature, FeatureLocation

تمثيل موقع اصطناعي (من القاعدة 100 إلى 120 على الشريط الموجب)

feature = SeqFeature(

```
    FeatureLocation(start=100, end=120, strand=+1),
    type="target_site",
    qualifiers={"note": "موقع مستهدف لتعديل CRISPR", "gene": "BRCA1"}
)
```

```
print(f"النوع: {feature.type}")
print(f"الموقع: {feature.location}")
print(f"المؤهلات: {feature.qualifiers}")
```

2. Coordinates and Strand

2. الإحداثيات (Coordinates) والاتجاه (Strand)

Understanding coordinates and strand is not just a technical detail. If you mis-specify the start or end, the feature may point to the wrong location on the

sequence. If you mis-specify the strand, you may interpret the feature backwards (e.g., reading a gene on the minus strand instead of the plus strand). This type of error can completely change the biological conclusion. For example, in designing a guide RNA for CRISPR, if you specify the target site on the wrong strand, you may design a guide that does not work, or worse, one that cuts at an off-target site.

3. Qualifiers and Annotations

فهم الإحداثيات والاتجاه ليس مجرد تفاصيل فنية. إذا أخطأت في تحديد البداية أو النهاية، فقد تشير الميزة إلى موقع خاطئ على التسلسل. وإذا أخطأت في الاتجاه، فقد تفسر الميزة بشكل معاكس (مثل قراءة جين على الشريط السالب بدلاً من الموجب). هذا النوع من الأخطاء يمكن أن يغير الاستنتاج البيولوجي بالكامل. مثلاً، في تصميم guide RNA لـ CRISPR، إذا حددت موقع الهدف على الشريط الخطأ، فقد تصمم دليلاً لا يعمل، أو الأسوأ، دليلاً يقطع في موقع غير مستهدف.

SeqFeature allows adding additional information via qualifiers. Qualifiers are a dictionary that stores descriptive information about the feature. In the CRISPR context, qualifiers may include:

3. المؤهلات (Qualifiers) والتعليقات التوضيحية (Annotations)

SeqFeature يسمح بإضافة معلومات إضافية عبر المؤهلات (qualifiers). المؤهلات هي قاموس (dictionary) يخزن معلومات وصفية عن الميزة. في سياق CRISPR، قد تشمل المؤهلات:

· guide_sequence: تسلسل guide RNA المصمم.

· gene: اسم الجين المستهدف.

· pam_sequence: تسلسل PAM المطلوب.

· gene: the name of the target gene.

· note: ملاحظات إضافية (مثل موقع القطع المتوقع، أو درجة التشابه مع مواقع غير مستهدفة).

· guide_sequence: the designed guide RNA sequence.

· off_target_candidates: قائمة بالمواقع غير المستهدفة المحتملة (كتمثيل تعليمي).

· pam_sequence: the required PAM sequence.

· note: additional notes (e.g., expected cut site, similarity to off-target sites).

· off_target_candidates: a list of potential off-target sites (as an educational representation).

```
# ميزة مع مؤهلات متعددة (تمثيل تعليمي لموقع CRISPR)
crispr_feature = SeqFeature(
```

```
    FeatureLocation(start=100, end=120, strand=+1),
    type="crispr_target",
    qualifiers={
        "gene": "BRCA1",
        "guide_sequence": "GACTAGCTAGCTAGCTAGC",
        "pam_sequence": "NGG",
        "note": "موقع مستهدف مع PAM في النهاية",
        "off_target_candidates": ["chr1:200-220", "chrX:1500-1520"]
    }
)

for key, value in crispr_feature.qualifiers.items():
    print(f"{key}: {value}")
```

Qualifiers make the feature rich in information, but they remain a representation of information, not proof of biological truth. Adding a qualifier like "off_target_candidates" does not mean these sites will actually be cut; it is just a candidate list for further study.

المؤهلات تجعل الميزة غنية بالمعلومات، لكنها تبقى تمثيلاً للمعلومة، وليس دليلاً على الحقيقة البيولوجية. إضافة مؤهل "off_target_candidates" لا يعني أن هذه المواقع ستُقطع فعلاً؛ إنها مجرد قائمة مرشحة للدراسة.

4. Coordinates in Biopython vs. GenBank: A Common Source of Errors

4. الإحداثيات في Biopython مقابل GenBank: مصدر شائع للأخطاء

One of the most common sources of errors in bioinformatics is confusion between different coordinate systems. Biopython uses zero-based coordinates with exclusive end, while GenBank uses one-based coordinates with inclusive end. When reading a GenBank file using Biopython, coordinates are automatically converted. However, if the workflow is writing coordinates manually, or converting between different formats, you must be aware of this difference.

أحد أكثر مصادر الأخطاء شيوعاً في المعلوماتية الحيوية هو الخلط بين أنظمة الإحداثيات المختلفة. Biopython تستخدم إحداثيات تبدأ من الصفر (zero-based) وتكون النهاية غير مشمولة (exclusive)، بينما GenBank تستخدم إحداثيات تبدأ من الواحد (one-based) وتكون النهاية مشمولة (inclusive). عند قراءة ملف GenBank باستخدام Biopython، يتم تحويل الإحداثيات تلقائياً. لكن إذا كنت تكتب إحداثيات يدوياً، أو تحول بين صيغ مختلفة، يجب أن تكون على دراية بهذا الاختلاف.

Language / نوع الكود: **Text / Coordinate Convention**

مثال: موقع يشمل القواعد من 5 إلى 10 (ضمنياً)

في - Biopython: start=5, end=11 (لأن end غير مشمولة)

في - GenBank: start=6, end=11 (بسبب الإحداثيات التي تبدأ من 1 والنهاية المشمولة)

```
# مثال
gb_start, gb_end = 6, 11
bp_start, bp_end = genbank_to_biopython(gb_start, gb_end)
print(f"GenBank: {gb_start}-{gb_end} -> Biopython: {bp_start}-{bp_end}")
# التحويل بين الأنظمة (مثال توضيحي)
def genbank_to_biopython(gb_start, gb_end):
    """تحويل إحداثيات GenBank (1-based, inclusive) إلى Biopython (0-based, exclusive)."""
    biopython_start = gb_start - 1
    biopython_end = gb_end # لأن النهاية مشمولة في GenBank
    return biopython_start, biopython_end
```

الدوال المعرفة هنا (genbank_to_biopython) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة.

القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة

للاختبار وإعادة الاستخدام.

5. Limits of Representation: The Feature Is Not Biological Proof

The chapter emphasizes that SeqFeature is a representation of information, not a practical design or biological proof. Adding a feature to a SeqRecord does not mean that this feature is biologically correct, or that it represents a real gene, or that it is ready for use in gene editing. It is merely a document saying "according to the available information, this location represents X." Moving from this representation to practical use (e.g., designing a guide RNA for gene editing) requires additional steps and independent verification. In the educational context, we use SeqFeature to teach how to represent an abstract region on a sequence record: start, end, strand, type, and description. This skill is essential because many bioinformatics errors arise from mixing up coordinates or strand orientations.

5. حدود التمثيل: الميزة ليست دليلاً بيولوجياً

6. Documenting the Reference Sequence

الفصل يؤكد على أن SeqFeature هو تمثيل للمعلومة، وليس تصميماً عملياً أو دليلاً بيولوجياً. إضافة ميزة إلى SeqRecord لا تعني أن هذه الميزة صحيحة بيولوجياً، أو أنها تمثل جيناً حقيقياً، أو أنها جاهزة للاستخدام في تحرير الجينات. هي مجرد وثيقة تقول "وفقاً للمعلومات المتوفرة، هذا الموقع يمثل كذا". الانتقال من هذا التمثيل إلى الاستخدام العملي (مثل تصميم guide RNA لتعديل جيني) يتطلب خطوات إضافية وتحققاً مستقلاً. في السياق

التعليمي، نستخدم SeqFeature لتعليم كيفية تمثيل منطقة افتراضية على سجل تسلسل: بداية، نهاية، اتجاه، نوع، ووصف. هذه المهارة أساسية لأن كثيراً من أخطاء المعلوماتية الحيوية تأتي من خلط الإحداثيات أو الاتجاهات.

When representing a location using SeqFeature, you must document the reference sequence you relied on. The reference is the exact version of the genome or sequence you used to determine the coordinates. In bioinformatics, reference versions change over time (e.g., hg19 vs. hg38 for the human genome), and coordinates may differ between versions. If you do not document the reference, no one will be able to accurately reconstruct the project's analysis.

6. توثيق التسلسل المرجعي

عند تمثيل موقع باستخدام SeqFeature، يجب أن توثق التسلسل المرجعي الذي اعتمدت عليه. المرجع هو الإصدار الدقيق للجينوم أو التسلسل الذي استخدمته لتحديد الإحداثيات. في المعلوماتية الحيوية، تتغير الإصدارات المرجعية بمرور الوقت (مثل hg19 مقابل hg38 للجينوم البشري)، والإحداثيات قد تختلف بين الإصدارات. إذا لم توثق المرجع، فلن يستطيع أحد إعادة بناء تحليلك بدقة.

```
# مثال: توثيق المرجع في المؤهلات
record.annotations["reference"] = {
"genome": "GRCh38",
"version": "hg38",
"date": "2025-01-15",
"source": "NCBI"
}
```

7. Practical Application: Adding Features to a SeqRecord

Let us apply what we have learned in a practical example: we have a hypothetical gene sequence, and we want to add features representing the gene location, the PAM site, and the expected cut site.

7. التطبيق العملي: إضافة ميزات إلى SeqRecord

لنطبق ما تعلمناه في مثال عملي: لدينا تسلسل جين وهمي، ونريد إضافة ميزات تمثل موقع الجين، موقع PAM، وموقع القطع المتوقع.

```
from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio.SeqFeature import SeqFeature, FeatureLocation
تسلسل وهمي #
```

gene_seq = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC" * 10)
record = SeqRecord(gene_seq, id="SYN_GENE_001", description="جين وهمي للتمرين")

# إضافة ميزة الجين (المنطقة الكاملة)
gene_feature = SeqFeature(
    FeatureLocation(start=0, end=300, strand=+1),
    type="gene",
    qualifiers={"gene": "SYN_GENE_001", "note": "جين وهمي للتمرين على SeqFeature"}
)
record.features.append(gene_feature)

# إضافة ميزة PAM (موقع NGG)
pam_feature = SeqFeature(
    FeatureLocation(start=40, end=43, strand=+1),
    type="PAM_site",
    qualifiers={"pam_sequence": "NGG", "note": "موقع PAM بالقرب من الهدف"}
)
record.features.append(pam_feature)

# إضافة ميزة هدف CRISPR (المكان الذي سيقطعه Cas9)
target_feature = SeqFeature(
    FeatureLocation(start=20, end=40, strand=+1),
    type="crispr_target",
    qualifiers={
        "guide_sequence": "GACTAGCTAGCTAGCTAGC",
        "cut_site": "ثلاث قواعد قبل PAM",
        "note": "موقع مستهدف لتعديل CRISPR"
    }
)
record.features.append(target_feature)

# عرض جميع الميزات
for feature in record.features:

Chapter Summary

خلاصة الفصل

SeqFeature is Biopython's tool for representing locations and features on a sequence. It uses coordinates (start and end), strand (plus or minus), type, and qualifiers to document the region and its biological significance. Accurate understanding of coordinates and strand is essential, because errors in them change the meaning of the feature. The difference between Biopython's (0-based, exclusive) and GenBank's (1-based, inclusive) systems is a common source of errors that must be noted. Features make the record rich in information, but they

remain a representation of information, not proof of biological truth. Documenting the reference sequence is an essential part of reproducibility. The next chapter will move to off-target analysis as a safety function in the CRISPR context.

SeqFeature هو أداة Biopython لتمثيل المواقع والميزات على التسلسل. يستخدم الإحداثيات (بداية ونهاية)، والاتجاه (شريط موجب أو سالب)، والنوع، والمؤهلات لتوثيق المنطقة وأهميتها البيولوجية. فهم الإحداثيات والاتجاه بدقة هو أساسي، لأن الأخطاء فيهما تغير معنى الميزة. الفصل بين نظامي Biopython (0-based، exclusive) و GenBank (1-based، inclusive) هو مصدر شائع للأخطاء يجب الانتباه له. الميزات تجعل السجل غنياً بالمعلومات، لكنها تبقى تمثيلاً للمعلومة وليس دليلاً على الحقيقة البيولوجية. توثيق التسلسل المرجعي هو جزء أساسي من إعادة الإنتاج. الفصل القادم سينتقل إلى تحليل المواقع غير المستهدفة (Off-target analysis) كوظيفة وقائية في سياق CRISPR.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Representing Features with SeqFeature" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "تمثيل المواقع باستخدام SeqFeature" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "adding a feature to a SeqRecord proves the existence of a real gene in the sequence." This is incorrect; the feature is a representation of the location according to available information, but it does not prove that this location is a real gene. The correct statement is: "SeqFeature represents a documented location on the sequence with information about its type and properties, but it requires independent evidence (such as database matching or experiments) to confirm its biological identity." A simple test: Practical check: "If this feature is in the record, does that mean the gene actually exists in the cell?".

مثال على التجاوز: القول إن "إضافة ميزة في SeqRecord تثبت وجود جين حقيقي في التسلسل". هذا غير صحيح؛ الميزة هي تمثيل للموقع وفقاً للمعلومات المتوفرة، لكنها لا تثبت أن هذا الموقع هو جين حقيقي. الصحيح هو: "SeqFeature يمثل موقعاً موثقاً على التسلسل مع معلومات عن نوعه وخصائصه، لكنه يحتاج إلى أدلة مستقلة (مثل مطابقة مع قاعدة بيانات أو تجارب) لتأكيد هويته البيولوجية." اختبار بسيط: فحص تطبيقي: "إذا كانت هذه الميزة موجودة في السجل، فهل هذا يعني أن الجين موجود فعلاً في الخلية؟".

Example: Representing a PAM site as a SeqFeature with the qualifier pam_sequence: "NGG" is a computational representation. Claiming that "this site is a real PAM site in the genome" is a biological claim not proven by the

representation alone. The representation tells us that the user marked this location as a PAM, but it does not prove that Cas9 will actually recognize it in the cell. Practical test: before relying on a feature in a SeqRecord, verify its source (e.g., a reference database) and confirm it with external references.

مثال: تمثيل موقع PAM كـ SeqFeature مع مؤهل "pam_sequence: "NGG هو تمثيل حسابي. الادعاء بأن "هذا الموقع هو موقع PAM حقيقي في الجينوم" هو ادعاء بيولوجي لا يثبته التمثيل وحده. التمثيل يخبرنا أن المستخدم وضع علامة على هذا الموقع باعتباره PAM، لكنه لا يثبت أن Cas9 سيتعرف عليه فعلاً في الخلية. اختبار عملي: قبل أن تعتمد على ميزة في SeqRecord، تحقق من مصدرها (مثل قاعدة بيانات مرجعية) وتأكد من صحتها بمصادر خارجية.

مثال تطبيقي على SeqFeature:

Applied example of SeqFeature:

عند إضافة ميزات إلى SeqRecord، يجب تسجيل ما يلي:

Chapter 30 — Off-Target Analysis as a Safety Function

الفصل 30 — Off-target analysis كوظيفة وقائية

الفكرة المركزية: مرشحات التشابه، السياق الجينومي، false positives/negatives، المراجعة.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

تحليل off-target في هذا السياق وظيفة فحص ومراجعة، لا وعداً بالسلامة. الخوارزمية قد تبحث عن مواقع مشابهة وفق معيار معين، لكن النتيجة تتأثر بقاعدة البيانات، والمعلمات، وحدود النموذج، ولا تختزل جميع المخاطر البيولوجية.

Off-target analysis is treated here as a screening and review function, not a guarantee of safety. An algorithm may search for similar sites under defined criteria, but the result depends on databases, parameters, and model limits and does not capture every biological risk.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is Off-Target Analysis؟

1. ما هو تحليل المواقع غير المستهدفة؟

Off-target analysis is a computational process aimed at identifying sites in the genome that the Cas9-guide RNA complex might cut unintentionally. This process relies on comparing the guide RNA sequence with the reference genome, searching for sites with high similarity (not perfect identity) to the primary target. The higher the similarity, the greater the likelihood that Cas9 will cut at that site. But similarity is not the only factor; genomic context (such as PAM presence, chromatin packaging, and accessibility) also affects the actual likelihood of cutting.

تحليل المواقع غير المستهدفة هو عملية حسابية تهدف إلى تحديد المواقع في الجينوم التي قد يقطعها مركب Cas9-guide RNA بشكل غير مقصود. تعتمد هذه العملية على مقارنة تسلسل guide RNA مع الجينوم المرجعي، والبحث عن مواقع ذات تشابه عالٍ (وليس تطابقاً تاماً) مع الهدف الأساسي. كلما زاد التشابه، زاد احتمال أن يقطع Cas9 هذا الموقع. لكن التشابه ليس العامل الوحيد؛ السياق الجينومي (مثل وجود PAM، وتغليف الكروماتين، وإمكانية الوصول) يؤثر أيضاً على احتمالية القطع الفعلية.

مثال مبسط:

الهدف الأساسي: GACTAGCTAGCTAGCTAGC

موقع غير مستهدف 1: GACTAGCTAGCTAGCTAGC (تطابق تام ← خطر مرتفع)

موقع غير مستهدف 2: GACTAGCTAGCTAGCTATC (اختلاف واحد ← خطر متوسط)

موقع غير مستهدف 3: GACTAGCTAGCTAGGTAGC (اختلافان ← خطر منخفض)

But even if similarity is high, cutting may not actually occur if the site is inaccessible due to chromatin packaging or other cellular factors. Therefore, off-target analysis is a candidate generation process for review, not a definitive prediction of activity.

2. Similarity ≠ Activity

لكن حتى لو كان التشابه مرتفعاً، قد لا يحدث القطع فعلاً إذا كان الموقع غير قابل للوصول بسبب تغليف الكروماتين أو عوامل خلوية أخرى. لذلك، تحليل المواقع غير المستهدفة هو عملية توليد مرشحين للمراجعة، وليس تنبؤاً قطعياً بالنشاط.

2. التشابه ≠ النشاط

A common mistake is assuming that any site with high similarity to the guide RNA will definitely be cut. This is incorrect. The actual activity of Cas9 depends on multiple factors beyond textual similarity:

من الأخطاء الشائعة افتراض أن أي موقع يحمل تشابهاً عالياً مع guide RNA سيُقطع حتماً. هذا غير صحيح. النشاط الفعلي لـ Cas9 يعتمد على عوامل متعددة تتجاوز التشابه النصي:

· توفر PAM: يجب أن يكون هناك تسلسل PAM مناسب (عادة NGG) بجوار الموقع. بدون PAM، لا يستطيع Cas9 الارتباط.

· PAM availability: there must be an appropriate PAM sequence (usually NGG) adjacent to the site. Without PAM, Cas9 cannot bind.

· إمكانية الوصول إلى الكروماتين: بعض مناطق الجينوم تكون مكثفة (مغلقة) وغير قابلة للوصول لـ Cas9.

· Chromatin accessibility: some genomic regions are condensed (closed) and inaccessible to Cas9.

· تعديلات الـ DNA: مثل المثيلة، قد تؤثر على الارتباط.

· DNA modifications: such as methylation, may affect binding.

· عوامل خلوية: وجود بروتينات أخرى قد تمنع أو تساعد في الارتباط.

· Cellular factors: the presence of other proteins may prevent or assist binding.

مثال:

موقع X: تشابه 90% مع الهدف، لكن لا يوجد PAM ← لا يُقطع.

موقع Y: تشابه 85% مع الهدف، يوجد PAM، الكروماتين مفتوح ← قد يُقطع.

For this reason, similarity should be treated as an initial filter, not as a final judgment. Computational outputs should be presented as a list for human review, with documented limitations and assumptions.

3. Building Outputs for Human Review

لهذا السبب، يجب التعامل مع التشابه كمرشح أولي، وليس كحكم نهائي. المخرجات الحسابية يجب أن تُقدّم كقائمة للمراجعة البشرية، مع توثيق القيود والافتراضات.

In the educational and security context, off-target analysis outputs should be built to be human-reviewable. This means:

3. بناء مخرجات للمراجعة البشرية

في السياق التعليمي والأمني، يجب أن تُبنى مخرجات تحليل المواقع غير المستهدفة بحيث تكون قابلة للمراجعة البشرية. هذا يعني:

· تحديد المواقع: عرض المواقع المحتملة مع إحداثياتها الدقيقة.

· درجة التشابه: حساب عدد الاختلافات بين guide RNA وكل موقع.

· Site identification: displaying potential sites with exact coordinates.

· وجود PAM: تحديد ما إذا كان الموقع يحتوي على PAM صالح.

· Similarity score: calculating the number of differences between the guide RNA and each site.

· السياق الجينومي: معلومات إضافية مثل الجين الذي يقع فيه الموقع، أو المناطق التنظيمية القريبة.

· PAM presence: determining whether the site contains a valid PAM.

· توصية: اقتراح إجراء (مثل مراجعة إضافية، أو اختبار معملي، أو استبعاد الدليل).

· Genomic context: additional information such as the gene in which the site is located, or nearby regulatory regions.

· Recommendation: suggesting an action (such as further review, laboratory testing, or guide exclusion).

مثال لمخرجات قابلة للمراجعة:

الموقع: chr1:12345-12365

الشريط: +

التشابه: 90% (اختلافان عن الهدف)

PAM: NGG (موجود)

الجين: BRCA1 (إكسون 2)

التوصية: مراجعة إضافية مطلوبة

This type of output allows the researcher (or the ethics committee) to make an informed decision, rather than relying on an automated decision that may be wrong.

4. Limits of Computational Analysis

هذا النوع من المخرجات يسمح للباحث (أو للجنة الأخلاقية) باتخاذ قرار مستنير، بدلاً من الاعتماد على قرار آلي قد يكون خاطئاً.

Off-target analysis has limits that must be clear:

4. حدود التحليل الحسابي

تحليل المواقع غير المستهدفة له حدود يجب أن تكون واضحة:

· الإيجابيات الكاذبة (False Positives): قد يحدد التحليل مواقع لن تُقطع فعلاً بسبب السياق الخلوي.

· السلبيات الكاذبة (False Negatives): قد يفوت التحليل مواقع تُقطع فعلاً بسبب عدم كمال خوارزميات البحث أو تغييرات في الجينوم المرجعي.

· False Positives: the analysis may identify sites that will not actually be cut due to cellular context.

· الاعتماد على المرجع: التحليل يعتمد على الجينوم المرجعي المستخدم. إذا كان المرجع مختلفاً عن جينوم الكائن الحي الفعلي (بسبب الاختلافات الطبيعية)، فقد تفوّت مواقع مهمة.

· False Negatives: the analysis may miss sites that are actually cut due to imperfect search algorithms or changes in the reference genome.

· عدم اليقين البيولوجي: حتى مع أفضل تحليل حسابي، لا يمكن التنبؤ بدقة بسلوك Cas9 في الخلية الحية.

· Reference dependence: the analysis depends on the reference genome used. If the reference differs from the actual organism's genome (due to natural variation), important sites may be missed.

· Biological uncertainty: even with the best computational analysis, the behavior of Cas9 in the living cell cannot be predicted with certainty.

For this reason, off-target analysis should be part of a broader process that includes human review, laboratory testing, and ethical evaluation.

5. Practical Application: Educational Simulation of Off-Target Analysis

لهذا السبب، يجب أن يكون تحليل المواقع غير المستهدفة جزءاً من عملية أوسع تشمل المراجعة البشرية، والاختبارات المعملية، والتقييم الأخلاقي.

In the educational context, off-target analysis can be simulated using synthetic data, without accessing sensitive operational databases. The goal is to teach principles, not to perform real analysis.

5. التطبيق العملي: محاكاة تعليمية لتحليل المواقع غير المستهدفة

في السياق التعليمي، يمكن محاكاة تحليل المواقع غير المستهدفة باستخدام بيانات اصطناعية، دون الوصول إلى قواعد بيانات تشغيلية حساسة. الهدف هو تعليم المبادئ، وليس تنفيذ تحليل حقيقي.

```
# محاكاة تعليمية لتحليل المواقع غير المستهدفة
def simulate_off_target_analysis(guide_rna, genome_subset, pam="NGG"):
    """
```

محاكاة بسيطة لتحديد المواقع غير المستهدفة في مجموعة فرعية من الجينوم.

للاستخدام التعليمي فقط (بيانات اصطناعية).

```
    """
    results = []
    for position, sequence in genome_subset.items():
        # حساب التشابه (عدد التطابقات)
        matches = sum(1 for a, b in zip(guide_rna, sequence) if a == b)
        similarity = matches / len(guide_rna)
```

```
                results.append({
                    "position": position,
                    "sequence": sequence,
                    "similarity": similarity,
                    "pam_present": pam_present,
                    "review_needed": similarity > 0.8 and pam_present
                })
```

```
# التحقق من وجود PAM (محاكاة بسيطة)
        pam_present = sequence.endswith(pam)  # تبسيط تعليمي
    return results
# بيانات اصطناعية
synthetic_genome = {
```

```
        "chr1:100-120": "GACTAGCTAGCTAGCTAGCNGG",
        "chr1:200-220": "GACTAGCTAGCTAGCTATCNGG",
        "chr1:300-320": "GACTAGCTAGCTAGGTAGCNGG",
        "chr2:100-120": "GACTAGCTAGCTAGCTAGCNGG",
        "chrX:500-520": "GACTAGCTAGCTAGCTAGCNNN",
    }
```

```
guide = "GACTAGCTAGCTAGCTAGC"
results = simulate_off_target_analysis(guide, synthetic_genome)
# عرض النتائج
```

print("نتائج تحليل المواقع غير المستهدفة (محاكاة تعليمية):")

```
for r in results:
    print(f"الموقع: {r['position']}, التشابه: {r['similarity']:.2f}, PAM: {r['pam_present']}, مراجعة: {r['review_needed']}")
```

الدوال المعرفة هنا (simulate_off_target_analysis) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational simulation illustrates the principle: identifying similar sites, evaluating PAM presence, and generating review recommendations. However, it does not replace real analysis or laboratory testing.

6. From Analysis to Decision: Human Review and Policies

هذه المحاكاة التعليمية توضح المبدأ: تحديد المواقع المشابهة، وتقييم وجود PAM، وتوليد توصيات للمراجعة. لكنها لا تغني عن التحليل الحقيقي أو الاختبارات المعملية.

Off-target analysis is not an end in itself; it is part of a broader decision process. After obtaining the list of potential sites, it should undergo a human review process that includes:

6. من التحليل إلى القرار: المراجعة البشرية والسياسات

تحليل المواقع غير المستهدفة ليس هدفاً في حد ذاته؛ هو جزء من عملية قرار أوسع. بعد الحصول على قائمة المواقع المحتملة، يجب أن تمر بعملية مراجعة بشرية تتضمن:

· تقييم المخاطر: ما هو احتمال أن يسبب القطع في هذا الموقع ضرراً؟

· Risk assessment: what is the likelihood that cutting at this site will cause harm?

· السياق السريري: هل هذا الموقع في جين معروف بالمرض، أم في منطقة غير مشفرة؟

· Clinical context: is this site in a gene known to be associated with disease, or in a non-coding region?

· التوصيات: هل يجب تعديل الدليل، أو استخدام دليل بديل، أو إجراء اختبارات إضافية؟

· Recommendations: should the guide be modified, an alternative guide used, or additional tests performed?

· التوثيق: تسجيل القرارات والأساس المنطقي لها (لمنع التكرار وضمان الشفافية).

· Documentation: recording decisions and the rationale behind them (to prevent repetition and ensure transparency).

In high-consequence systems, the default path should be to stop or escalate to human review in the absence of explicit authorization or trusted reference (the fail-closed principle). This ensures that decisions are not fully automated, but subject to human accountability.

في الأنظمة ذات العواقب العالية، يجب أن يكون المسار الافتراضي هو التوقف أو التصعيد إلى مراجعة بشرية عند غياب الإذن الصريح أو المرجع الموثوق (مبدأ fail-closed). هذا يضمن أن القرارات ليست آلية تماماً، بل تخضع للمساءلة البشرية.

Chapter Summary

خلاصة الفصل

Off-target analysis is a safety function in the CRISPR context, aimed at identifying potential sites of unintentional cutting. It relies on comparing guide RNA similarity with the reference genome, but similarity is not the sole indicator; genomic context and cellular factors affect actual activity. Therefore, similarity should be treated as an initial filter, and outputs should be built for human review. False positives and false negatives are inherent limits of computational analysis, requiring laboratory testing and independent verification. In the educational context, analysis can be simulated using synthetic data to teach principles without handling sensitive data. The final decision should undergo human review, with documentation of decisions and rationale, and adherence to the fail-closed principle in critical systems. The next chapter will move to bioethics and algorithmic biosecurity.

تحليل المواقع غير المستهدفة هو وظيفة وقائية في سياق CRISPR، تهدف إلى تحديد المواقع المحتملة للقطع غير المقصود. يعتمد على مقارنة تشابه guide RNA مع الجينوم المرجعي، لكن التشابه ليس المؤشر الوحيد؛ السياق الجينومي والعوامل الخلوية تؤثر على النشاط الفعلي. لذلك، يجب التعامل مع التشابه كمرشح أولي، وبناء مخرجات قابلة للمراجعة البشرية. الإيجابيات الكاذبة والسلبيات الكاذبة حدود متأصلة في التحليل الحسابي، وتتطلب اختبارات معملية وتحققاً مستقلاً. في السياق التعليمي، يمكن محاكاة التحليل باستخدام بيانات اصطناعية لتعليم المبادئ دون التعامل مع بيانات حساسة. القرار النهائي يجب أن يمر بمراجعة بشرية، مع توثيق القرارات والأساس المنطقي لها، والتزام بمبدأ fail-closed في الأنظمة الحرجة. الفصل القادم سينتقل إلى الأخلاقيات الحيوية والسلامة الجينية البرمجية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Off-Target Analysis as a Safety Function" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Off-target analysis كوظيفة وقائية" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "off-target analysis can predict all cut sites with 100% accuracy." This is incorrect; the analysis identifies candidates, but actual activity depends on cellular factors that cannot be computationally predicted. The correct statement is: "off-target analysis prioritizes potential sites for unintended cutting, but it requires human review and laboratory testing to

verify actual activity." A simple test: Practical check: "Does this analysis guarantee that the identified sites will actually be cut, or are they just candidates for review?".

مثال على التجاوز: القول إن "تحليل المواقع غير المستهدفة يمكنه التنبؤ بجميع مواقع القطع بدقة 100%". هذا غير صحيح؛ التحليل يحدد مرشحين، لكن النشاط الفعلي يعتمد على عوامل خلوية لا يمكن التنبؤ بها حسابياً. الصحيح هو: "تحليل المواقع غير المستهدفة يرشح مواقع محتملة للقطع غير المقصود، لكنه يتطلب مراجعة بشرية واختبارات معملية للتحقق من النشاط الفعلي." اختبار بسيط: فحص تطبيقي: "هل هذا التحليل يضمن أن المواقع المحددة ستُقطع فعلاً، أم أنها مجرد مرشحين للمراجعة؟"ـ

Example: The list of potential off-target sites is a computational representation. Claiming that "these sites will definitely be cut in the cell" is a biological claim not justified by the computational analysis alone. The analysis identifies textual similarity, but does not predict actual activity in the cellular context. Practical test: before inferring the risk of an off-target site from a computational analysis, look for independent evidence (such as in vitro experiments, or published studies) supporting this conclusion.

مثال: قائمة المواقع غير المستهدفة المحتملة هي تمثيل حسابي. الادعاء بأن "هذه المواقع ستُقطع بالتأكيد في الخلية" هو ادعاء بيولوجي غير مبرر بالتحليل الحسابي وحده. التحليل يحدد التشابه النصي، لكنه لا يتنبأ بالنشاط الفعلي في السياق الخلوي. اختبار عملي: قبل أن تستنتج خطورة موقع غير مستهدف من تحليل حسابي، ابحث عن أدلة مستقلة (مثل تجارب في المختبر، أو دراسات منشورة) تدعم هذا الاستنتاجـ

مثال تطبيقي على تحليل المواقع غير المستهدفة:

· المدخلات: تسلسل guide RNA، جينوم مرجعي (مع الإصدار)، تسلسل PAM المطلوب، عتبة التشابهـ

· الافتراضات: أن الجينوم المرجعي يمثل الجينوم الفعلي للكائن الحي، وأن التسلسلات المتشابهة قد تكون مواقع قطع محتملة، وأن عتبة التشابه المختارة مناسبةـ

· المخرجات: قائمة بالمواقع المحتملة مع درجات التشابه، وجود PAM، وسياق جينومي.

· القيود: التحليل لا يأخذ في الاعتبار تغليف الكروماتين، أو تعديلات الـ DNA، أو العوامل الخلوية الأخرى التي تؤثر على النشاط الفعلي. قد يكون هناك إيجابيات كاذبة وسلبيات كاذبةـ

Applied example of off-target analysis:

اختبار: قبل اعتماد نتائج التحليل، تحقق من عتبة التشابه المستخدمة، وجود PAM في المواقع المحددة، والسياق الجينومي (هل الموقع في جين مهم؟)ـ

· Inputs: guide RNA sequence, reference genome (with version), required PAM sequence, similarity threshold.

· Assumptions: that the reference genome represents the actual organism's genome, that similar sequences are potential cut sites, and that the chosen similarity threshold is appropriate.

· Outputs: a list of potential sites with similarity scores, PAM presence, and genomic context.

· Limitations: the analysis does not account for chromatin packaging, DNA modifications, or other cellular factors affecting actual activity. There may be false positives and false negatives.

Test: before relying on the analysis results, check the similarity threshold used, PAM presence at the identified sites, and genomic context (is the site in an important gene?).

عند إجراء تحليل المواقع غير المستهدفة، يجب تسجيل ما يلي:

· المصدر: التسلسل المستهدف والجينوم المرجعي المستخدم (الإصدار والتاريخ).

· الإصدار: إصدار الأداة المستخدمة، وإصدار قاعدة البيانات المرجعية.

· المعلمات: عتبة التشابه، تسلسل PAM المطلوب، أي معاملات أخرى.

· التحويلات: أي خطوات لتنقية البيانات أو تحويلها قبل التحليل.

When performing off-target analysis, record the following:

· Provenance: the target sequence and the reference genome used (version and date).

· Version: the tool version used, and the reference database version.

· Parameters: the similarity threshold, required PAM sequence, and any other parameters.

· Transformations: any steps to clean or transform data before analysis.

Test: Practical check: "If I gave this record to a colleague a year later, could they re-run the analysis and obtain the same results without my help?" If the answer is no, the documentation is insufficient.

قبل أن تعتمد على أداة تحليل المواقع غير المستهدفة في مشروع حقيقي، اختبر الأداة على بيانات اصطناعية ذات مواقع معروفة. أنشئ جينوماً اصطناعياً صغيراً يحتوي على هدف رئيسي وعدة مواقع مشابهة. شغّل الأداة وتأكد من

أنها تكتشف المواقع المتوقعة (حساسية ونوعية). إذا نجح الاختبار، فأنت واثق من أن الأداة تعمل كما هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس خصائص الجينوم وليس خطأ في الأداة.

Before relying on an off-target analysis tool in a real project, test the tool on synthetic data with known sites. Create a small synthetic genome containing a primary target and several similar sites. Run the tool and verify that it detects the expected sites (sensitivity and specificity). If the test succeeds, the workflow is confident that the tool works as expected, and any result with real data reflects the genome's properties, not an error in the tool.

Part 9 — Bioethics and Algorithmic Biosecurity

الباب 9 — الأخلاقيات الحيوية والسلامة الجينية البرمجية

الفكرة المركزية: القيم والحقوق، الحوادث، سوء الاستخدام، الحوكمة.

Chapter 31 — Bioethics, Biosafety, and Biosecurity

الفصل 31 — Bioethics و Biosafety و Biosecurity

شرح أعمق للمفهوم

Deeper Conceptual Explanation

الأخلاقيات الحيوية تسأل ما الذي ينبغي فعله ولماذا، بينما تركز biosafety على تقليل الحوادث غير المقصودة، وbiosecurity على منع الوصول أو الاستخدام الضار. التمييز بينها ليس لغوياً؛ فهو يغير نوع الخطر ونوع الضابط الذي يجب تصميمه.

Bioethics asks what ought to be done and why, while biosafety focuses on reducing unintended harm and biosecurity on preventing harmful access or misuse. The distinction is not merely terminological; it changes the type of risk and the type of control that should be designed.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Bioethics

Bioethics is a branch of ethical philosophy concerned with moral questions arising from advances in biology and medicine. It asks about values, rights, and duties: what are the ethical limits of genetic modification? Who decides what is acceptable and what is not? How do we fairly distribute the benefits of biotechnologies?

Bioethics is not just a set of rules; it is a continuous process of critical thinking, involving multiple perspectives (scientific, religious, legal, social).

1. الأخلاقيات الحيوية (Bioethics)

الأخلاقيات الحيوية هي فرع من الفلسفة الأخلاقية يهتم بالأسئلة الأخلاقية الناشئة عن التقدم في البيولوجيا والطب.

تسأل عن القيم والحقوق والواجبات: ما هي الحدود الأخلاقية للتعديل الجيني؟ من يقرر ما هو مقبول وما هو غير مقبول؟ كيف نوزع فوائد التقنيات الحيوية بشكل عادل؟ الأخلاقيات الحيوية ليست مجرد مجموعة من القواعد؛ هي عملية تفكير نقدي مستمر، تشارك فيها وجهات نظر متعددة (علمية، دينية، قانونية، اجتماعية).

في سياق المعلوماتية الحيوية، الأخلاقيات الحيوية تطرح أسئلة مثل:

· هل يجوز استخدام بيانات جينومية دون موافقة صريحة من صاحبها؟

In the context of bioinformatics, bioethics raises questions such as:

· كيف نضمن الخصوصية والسرية في قواعد البيانات الجينية؟

· Is it permissible to use genomic data without explicit consent from its owner?

· من يملك المعلومات الجينية: الفرد، العائلة، المجتمع، أم الشركات؟

· How do we ensure privacy and confidentiality in genetic databases?

· كيف نمنع استخدام المعلومات الجينية للتمييز (في التأمين، العمل، التعليم)؟

· Who owns genetic information: the individual, the family, society, or companies?

· How do we prevent the use of genetic information for discrimination (in insurance, employment, education)?

مبدأ أساسي في الأخلاقيات الحيوية: الموافقة المستنيرة (Informed Consent).

لا يجوز استخدام عينة بيولوجية أو بيانات جينية من شخص دون إعلامه كامل بطبيعة البحث، ومخاطره، وفوائده، والحصول على موافقته الصريحة.

2. Biosafety

Biosafety focuses on preventing accidental exposure or unintended release of hazardous organisms or biological materials. It includes containment procedures, safe handling of hazardous materials, safety training, and biological waste management. In laboratories, organisms are classified by risk level (BSL-1 to BSL-4), and appropriate containment is required for each level. In the computational context, biosafety means ensuring that software systems do not inadvertently release information or sequences that could be used unsafely.

2. السلامة البيولوجية (Biosafety)

السلامة البيولوجية تركز على منع التعرض العرضي أو الإطلاق غير المقصود للكائنات الحية أو المواد البيولوجية الخطرة. تشمل إجراءات الاحتواء (containment)، والتعامل الآمن مع المواد الخطرة، والتدريب على السلامة، وإدارة النفايات البيولوجية. في المختبرات، تُصنف الكائنات الحية حسب مستوى الخطورة (BSL-1 إلى BSL-4)، ويُطلب مستوى احتواء مناسب لكل فئة. في السياق الحاسوبي، السلامة البيولوجية تعني التأكد من أن الأنظمة البرمجية لا تسبب عن غير قصد إطلاق معلومات أو تسلسلات يمكن استخدامها بشكل غير آمن.

مثال: في تحليل بيانات الجينوم، السلامة البيولوجية تعني:

- تخزين البيانات الحساسة في أنظمة آمنة.

- تقييد الوصول إلى التسلسلات التي قد تكون خطرة.

- التأكد من أن الأدوات البرمجية لا تنتج عن غير قصد تسلسلات يمكن تركيبها بشكل خطير.

3. Biosecurity

Biosecurity addresses protection against theft, unauthorized access, or deliberate misuse of biological materials and information. It differs from biosafety in that it focuses on intentional threats (such as bioterrorism, industrial espionage, or unethical use of biotechnologies). In the computational context, biosecurity means protecting systems and data from cyberattacks aimed at stealing, altering, or misusing genetic information.

3. الأمن البيولوجي (Biosecurity)

الأمن البيولوجي يتناول منع السرقة أو الوصول غير المصرح به أو سوء الاستخدام المتعمد للمواد البيولوجية والمعلومات. يختلف عن السلامة البيولوجية في أنه يركز على التهديدات المتعمدة (مثل الإرهاب البيولوجي، أو التجسس الصناعي، أو الاستخدام غير الأخلاقي للتقنيات الحيوية). في السياق الحاسوبي، الأمن البيولوجي يعني حماية الأنظمة والبيانات من الهجمات السيبرانية التي تهدف إلى سرقة أو تغيير أو إساءة استخدام المعلومات الجينية.

أمثلة على الأمن البيولوجي:

- منع الوصول غير المصرح به إلى قواعد بيانات الجينوم البشري الحساسة.

- تأمين الأنظمة التي تصمم تسلسلات DNA اصطناعية.

- مراقبة طلبات الحصول على تسلسلات خطرة محتملة.

4. Distinguishing the Three Concepts

Confusing bioethics, biosafety, and biosecurity is a common mistake that can lead to unbalanced decisions. Each concept has its own scope:

4. التمييز بين المفاهيم الثلاثة

الخلط بين الأخلاقيات الحيوية، السلامة البيولوجية، والأمن البيولوجي هو خطأ شائع يمكن أن يؤدي إلى قرارات غير

متوازنة. كل مفهوم له نطاقه الخاص:

المفهوم السؤال المحوري التركيز المثال.

Concept Core Question Focus Example.

الأخلاقيات الحيوية ما هو الصواب والخطأ؟ القيم والحقوق والمسؤوليات هل يجب تعديل جينات الأجنة؟

Bioethics What is right and wrong? Values, rights, responsibilities Should we edit embryo genomes?

السلامة البيولوجية كيف نمنع الحوادث؟ الاحتواء والتعامل الآمن كيفية التخلص الآمن من عينات ملوثة.

Biosafety How do we prevent accidents? Containment, safe handling How to safely dispose of contaminated samples.

الأمن البيولوجي كيف نمنع سوء الاستخدام المتعمد؟ الحماية من التهديدات منع سرقة مسببات الأمراض.

Biosecurity How do we prevent deliberate misuse? Protection from threats Preventing theft of pathogens.

Understanding these differences is essential for developing balanced policies. Ethics cannot be replaced by technical measures, nor can security risks be ignored under the guise of technical safety.

5. Layered Governance

فهم هذه الفروق ضروري لتطوير سياسات متوازنة. لا يمكن استبدال الأخلاقيات بإجراءات تقنية، ولا يمكن تجاهل

المخاطر الأمنية بحجة السلامة التقنية.

In sensitive systems, governance must be layered, combining:

5. الحوكمة متعددة الطبقات (Layered Governance)

في الأنظمة الحساسة، يجب أن تكون الحوكمة متعددة الطبقات، بحيث تجمع بين:

· Technical layer: software security measures (e.g., authentication, encryption, audit logs).

· الطبقة التقنية: إجراءات أمنية برمجية (مثل المصادقة، التشفير، سجلات التدقيق).

· Procedural layer: policies and procedures (e.g., prior approval, ethical review, contingency plans).

· الطبقة الإجرائية: سياسات وإجراءات (مثل الموافقة المسبقة، المراجعة الأخلاقية، خطط الطوارئ).

· Human layer: individual responsibility (e.g., training, ethical awareness, accountability).

· الطبقة البشرية: مسؤولية الأفراد (مثل التدريب، الوعي الأخلاقي، المحاسبة).

مثال على الحوكمة المتعددة الطبقات في مشروع تعديل جيني:

1. طبقة تقنية: نظام برمجي يرفض تنفيذ تعديلات غير مصرح بها (fail-closed).

2. طبقة إجرائية: يجب الحصول على موافقة لجنة أخلاقية قبل البدء.

3. طبقة بشرية: كل باحث مسؤول عن توثيق قراراته وأفعاله.

6. Human Responsibility in Automated Systems

Automation does not eliminate human responsibility. Even with the best algorithms, final decisions remain the responsibility of humans. In automated systems, there must be a mechanism for human review, especially for high-risk cases. This means the system should be designed to stop execution when a condition is not met (fail-closed), and escalate to human review when needed.

6. المسؤولية البشرية في الأنظمة المؤتمتة

الأتمتة لا تلغي المسؤولية البشرية. حتى مع أفضل الخوارزميات، تبقى القرارات النهائية مسؤولية البشر. في الأنظمة المؤتمتة، يجب أن تكون هناك آلية للمراجعة البشرية، خاصة للحالات عالية المخاطر. هذا يعني أن النظام يجب أن يكون مصمماً بحيث يوقف التنفيذ عند فقدان الشرط (fail-closed)، ويرفع الحالة إلى مراجعة بشرية عند الحاجة.

مبدأ أساسي: الأتمتة تدعم القرار، لا تحل محل القرار البشري.

7. Practical Application: Risk Assessment of a Hypothetical Project

Let's apply what we have learned to a hypothetical project: you want to edit a gene in human cells to treat a genetic disease. How do you assess risks from the perspectives of Bioethics, Biosafety, and Biosecurity?

7. التطبيق العملي: تقييم مخاطر مشروع وهمي

لنطبق ما تعلمناه على مشروع وهمي: تريد تعديل جين في خلايا بشرية لعلاج مرض وراثي. كيف تقيم المخاطر من منظور Bioethics، Biosafety، و Biosecurity؟

تقييم الأخلاقيات الحيوية (Bioethics):

- هل حصلت على موافقة مستنيرة من المرضى؟

- هل التعديل آمن طويل الأمد؟

- هل الفوائد تفوق المخاطر؟

- هل هناك عدالة في الوصول للعلاج؟

تقييم السلامة البيولوجية (Biosafety):

- هل المختبر مجهز للتعامل مع الخلايا البشرية المعدلة؟

- هل هناك إجراءات لمنع التلوث أو التسرب؟

- كيف تتخلص من المواد البيولوجية بأمان؟

تقييم الأمن البيولوجي (Biosecurity):

- من لديه حق الوصول إلى الخلايا والتسلسلات؟

- هل النظام البرمجي محمي ضد الاختراق؟

- هل هناك سياسات لمنع سوء استخدام التقنية؟

Chapter Summary

خلاصة الفصل

Bioethics, biosafety, and biosecurity are three fundamental but distinct concepts. Bioethics asks about values, rights, and responsibilities. Biosafety focuses on preventing accidents and accidental exposure. Biosecurity addresses prevention of deliberate misuse. Distinguishing between these concepts is essential for developing balanced policies. Governance must be layered, combining technical, procedural, and human measures. Automation does not eliminate human responsibility; it should support decision-making, not replace it. High-consequence systems should follow the fail-closed principle. The next chapter will move to Sequence Screening as a defensive tool.

الأخلاقيات الحيوية، السلامة البيولوجية، والأمن البيولوجي هي ثلاثة مفاهيم أساسية لكنها متميزة. الأخلاقيات الحيوية تسأل عن القيم والحقوق والمسؤوليات. السلامة البيولوجية تركز على منع الحوادث والتعرض العرضي. الأمن البيولوجي يتناول منع سوء الاستخدام المتعمد. الفصل بين هذه المفاهيم ضروري لتطوير سياسات متوازنة. الحوكمة

يجب أن تكون متعددة الطبقات، تجمع بين الإجراءات التقنية والإجرائية والبشرية. الأتمتة لا تلغي المسؤولية البشرية؛ يجب أن تدعم القرار، لا أن تحل محله. الأنظمة ذات العواقب العالية يجب أن تتبع مبدأ fail-closed. الفصل القادم سينتقل إلى Sequence Screening كأداة دفاعية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Bioethics, Biosafety, and Biosecurity" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Bioethics و Biosafety و Biosecurity" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "following biosafety procedures solves all ethical problems." This is incorrect; biosafety prevents accidents, but it does not answer ethical questions such as fairness or consent. The correct statement is: "biosafety, biosecurity, and bioethics are distinct but complementary concepts; each addresses a different aspect of risks and responsibilities in biological work, and none can replace the others." A simple test: Practical check: "Does this measure prevent an accident, protect against misuse, or address an ethical question?".

مثال على التجاوز: القول إن "اتباع إجراءات السلامة البيولوجية يحل جميع المشاكل الأخلاقية". هذا غير صحيح؛ السلامة البيولوجية تمنع الحوادث، لكنها لا تجيب عن الأسئلة الأخلاقية مثل العدالة أو الموافقة. الصحيح هو: "السلامة البيولوجية، الأمن البيولوجي، والأخلاقيات الحيوية هي مفاهيم متميزة متكاملة؛ كل منها يعالج جانبا مختلفا من المخاطر والمسؤوليات في العمل البيولوجي، ولا يمكن لأحدها أن يحل محل الآخر." اختبار بسيط: فحص تطبيقي: "هل هذا الإجراء يمنع حادثة، أم يحمي من سوء استخدام، أم يعالج سؤالاً أخلاقياً؟".

Example: A software system implementing a security policy is a computational representation of that policy. Claiming that "this system guarantees ethical behavior" is an ethical claim not proven by the system alone. The system can apply specific rules, but it cannot understand ethical context or balance conflicting values. Practical test: before considering a software system "ethical," verify that there is human review and accountability mechanisms.

مثال: نظام برمجي يطبق سياسة أمنية هو تمثيل حسابي للسياسة. الادعاء بأن "هذا النظام يضمن السلوك الأخلاقي" هو ادعاء أخلاقي لا يثبته النظام وحده. النظام يمكنه تطبيق قواعد محددة، لكنه لا يستطيع فهم السياق الأخلاقي أو الموازنة بين القيم المتضاربة. اختبار عملي: قبل أن تعتبر نظاماً برمجياً "أخلاقياً"، تحقق من أن هناك مراجعة بشرية وآليات للمساءلة.

مثال تطبيقي على نظام إدارة سياسات أمنية:

· المدخلات: سياسات أمنية محددة (مثل من يمكنه الوصول إلى أي بيانات)، وسجلات الوصول، وتصنيفات المخاطر.

· الافتراضات: أن السياسات تعكس القيم الأخلاقية والقانونية، وأن سجلات الوصول دقيقة، وأن تصنيف المخاطر صحيح.

· المخرجات: قرارات الوصول (قبول/رفض)، وسجلات تدقيق، وتنبيهات للحالات غير العادية.

· القيود: النظام لا يستطيع تقييم السياق الأخلاقي للمواقف غير المتوقعة، ولا يمكنه تعديل السياسات دون تدخل بشري.

Applied example of a security policy management system:

اختبار: قبل الاعتماد على نظام سياسات أمنية، تحقق من أن هناك آلية للمراجعة البشرية للحالات المستثناة، وأن النظام يسجل جميع القرارات لتتبعها.

· Inputs: specific security policies (e.g., who can access which data), access logs, and risk classifications.

· Assumptions: that the policies reflect ethical and legal values, that access logs are accurate, and that risk classifications are correct.

· Outputs: access decisions (accept/deny), audit logs, and alerts for unusual cases.

· Limitations: the system cannot evaluate the ethical context of unexpected situations, nor can it modify policies without human intervention.

Test: before relying on a security policy system, verify that there is a mechanism for human review of exceptional cases, and that the system logs all decisions for traceability.

عند تصميم أو استخدام أنظمة أمنية، يجب تسجيل:

· المصدر: السياسات والإجراءات التي يستند إليها النظام (المراجع القانونية والأخلاقية).

· الإصدار: إصدار النظام البرمجي، وإصدار السياسات المستخدمة.

· المعلمات: أي معاملات تؤثر على قرارات النظام (مثل عتبات المخاطر، قواعد الوصول).

· التحويلات: أي تحديثات أو تعديلات على السياسات أو النظام.

When designing or using security systems, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لمراجع بعد عام، هل يستطيع فهم كيف اتخذت القرارات، وما هي السياسات التي طُبقت، وهل كانت متسقة مع المعايير الأخلاقية والقانونية؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the policies and procedures the system is based on (legal and ethical references).

· Version: the software system version, and the version of policies used.

· Parameters: any parameters affecting system decisions (e.g., risk thresholds, access rules).

· Transformations: any updates or modifications to policies or the system.

Test: Practical check: "If I gave this record to a reviewer a year later, could they understand how decisions were made, what policies were applied, and whether they were consistent with ethical and legal standards?" If the answer is no, the documentation is insufficient.

قبل تطبيق أي نظام أمني أو أخلاقي على بيانات حقيقية، اختبره على سيناريوهات اصطناعية. أنشئ حالات اختبار تمثل مواقف أخلاقية وأمنية مختلفة (مثل طلب وصول من شخص غير مصرح له، أو محاولة تصدير بيانات حساسة). تحقق من أن النظام يتعامل مع كل حالة بشكل صحيح وفقاً للسياسات المحددة. إذا نجح الاختبار، فأنت واثق من أن النظام يعمل كما هو متوقع، وأي حالة مع بيانات حقيقية تعكس تعقيدات العالم الحقيقي التي قد تتطلب مراجعة بشرية.

Before applying any security or ethical system to real data, test it on synthetic scenarios. Create test cases representing different ethical and security situations (e.g., an access request from an unauthorized person, or an attempt to export sensitive data). Verify that the system handles each case correctly according to the specified policies. If the test succeeds, the workflow is confident that the system works as expected, and any case with real data reflects real-world complexities that may require human review.

Chapter 32 — Sequence Screening

الفصل

الفكرة المركزية: pipeline دفاعي، التصنيف، المراجعة البشرية، الإصدارات، التدقيق.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Sequence matching is one assessment tool rather than an independent verdict. In a defensive system, a match can trigger human review or safe stopping rather than become an automatic final decision. This reduces dependence on a single threshold and makes uncertainty part of the system design.

مطابقة التسلسل أداة من أدوات التقييم وليست حكماً مستقلاً. في نظام دفاعي، يمكن أن تكون نتيجة الفحص إشارة إلى مراجعة بشرية أو إلى إيقاف آمن، بدلاً من تحويل المطابقة إلى قرار نهائي آلي. هذا التصميم يقلل الاعتماد على عتبة واحدة ويجعل عدم اليقين جزءاً من النظام.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Sequence Screening as a Governance Pipeline

Sequence screening is not just a single command that runs and gives a result. It is an integrated pipeline that passes through several stages:

1. فحص التسلسل كخط أنابيب حوكمة

فحص التسلسل ليس مجرد أمر واحد يُشغل ويعطي نتيجة. هو خط أنابيب متكامل يمر بعدة مراحل:

المرحلة الأولى: إدخال البيانات (Data Input).

يُدخل تسلسل (أو مجموعة تسلسلات) إلى النظام. قد يكون التسلسل من مصدر خارجي (مثل طلب شراء من شركة تخليق) أو من داخل المؤسسة (مثل تسلسل مركب حديثاً). يجب أن يُسجل مصدر التسلسل، وتاريخه، والغرض منه.

المرحلة الثانية: الفحص الآلي (Automated Screening).

يقوم النظام بمقارنة التسلسل مع قواعد بيانات مرجعية محدثة. هذه المقارنة تبحث عن تشابه مع تسلسلات معروفة الخطورة (مثل مسببات الأمراض، أو السموم، أو عناصر محظورة). لكن الفحص الآلي ليس نهائياً؛ إنه خطوة أولى لتوليد مرشحين للمراجعة.

المرحلة الثالثة: تصنيف المخاطر (Risk Classification).

بناءً على نتائج الفحص، يُصنف التسلسل إلى فئات: منخفض المخاطر، متوسط المخاطر، مرتفع المخاطر. يعتمد التصنيف على درجة التشابه، والسياق، والسياسات المحددة.

المرحلة الرابعة: المراجعة البشرية (Human Review).

Stage One: Data Input

في الحالات متوسطة وعالية المخاطر، يجب أن يمر الطلب بمراجعة بشرية. يقوم مراجع (أو لجنة) بفحص النتائج، وتقييم السياق، واتخاذ قرار. هذه المراجعة هي الطبقة الأكثر أهمية في النظام، لأنها تسمح بالتعامل مع الحالات غير المتوقعة أو الملتبسة.

A sequence (or set of sequences) is entered into the system. The sequence may come from an external source (such as a purchase order from a synthesis

company) or from within the institution (such as a newly synthesized sequence). The source of the sequence, its date, and its purpose should be recorded.

المرحلة الخامسة: تسجيل القرار (Decision Recording).

Stage Two: Automated Screening

يُسجل كل قرار (قبول، رفض، تعليق) مع الأساس المنطقي له، وتاريخه، وهوية المراجع. هذا التسجيل هو ما يجعل النظام قابلاً للتدقيق والمراجعة لاحقاً.

The system compares the sequence with updated reference databases. This comparison looks for similarity with known hazardous sequences (such as pathogens, toxins, or prohibited elements). However, automated screening is not final; it is a first step to generate candidates for review.

Stage Three: Risk Classification.

Based on the screening results, the sequence is classified into categories: low risk, medium risk, high risk. The classification depends on the degree of similarity, context, and specific policies.

Stage Four: Human Review

In medium- and high-risk cases, the request must undergo human review. A reviewer (or committee) examines the results, evaluates the context, and makes a decision. This review is the most important layer of the system, as it allows handling unexpected or ambiguous cases.

Stage Five: Decision Recording

Every decision (accept, reject, hold) is recorded with its rationale, date, and reviewer identity. This recording is what makes the system auditable and reviewable later.

2. Why Is Automated Screening Not Enough؟

2. لماذا الفحص الآلي غير كافٍ؟

Automated screening relies on reference databases. These databases may be incomplete, outdated, or may not cover newly unclassified sequences. Also, textual similarity is not a sufficient indicator of biological activity. A sequence may be similar to a known toxin but not perform the same function due to differences in context or structure. Therefore, automated screening should be used as an initial filter, not as a final judgment.

الفحص الآلي يعتمد على قواعد بيانات مرجعية. هذه القواعد قد تكون غير مكتملة، أو قديمة، أو لا تغطي تسلسلات جديدة لم تُصنف بعد. أيضاً، التشابه النصي ليس مؤشراً كافياً للنشاط البيولوجي. قد يكون تسلسل مشابهاً لسم معروف لكنه لا يؤدي نفس الوظيفة بسبب اختلافات في السياق أو التركيب. لذلك، الفحص الآلي يجب أن يُستخدم كمرشح أولي، وليس كحكم نهائي.

مثال: تسلسل X مشابه لسم معين بنسبة 90%، لكنه يفتقر إلى موقع نشط حيوي.

الفحص الآلي: يصنفه كمرتفع المخاطر (بناءً على التشابه).

المراجعة البشرية: بعد فحص السياق، تقرر أنه منخفض المخاطر (لأن الموقع النشط مفقود).

3. Classification and Human Review

Classification is not just an automated process; it is a decision based on specific policies and human review. In high-consequence systems, the default path should be to stop or escalate to human review in the absence of explicit authorization or trusted reference. This principle (fail-closed) ensures that errors in automated screening do not lead to incorrect decisions.

3. التصنيف والمراجعة البشرية

التصنيف ليس مجرد عملية آلية؛ هو قرار يعتمد على سياسات محددة ومراجعة بشرية. في الأنظمة ذات العواقب العالية، يجب أن يكون المسار الافتراضي هو التوقف أو التصعيد إلى مراجعة بشرية عند غياب الإذن الصريح أو المرجع الموثوق. هذا المبدأ (fail-closed) يضمن أن الأخطاء في الفحص الآلي لا تؤدي إلى قرارات خاطئة.

مبدأ fail-closed في فحص التسلسل:

إذا لم يكن النظام متأكداً من تصنيف تسلسل (مثلاً، بسبب قاعدة بيانات قديمة)،

فالسلوك الافتراضي هو الرفض أو التصعيد للمراجعة، وليس القبول.

Human review is not an obstacle to science; it is a way to make decisions accountable and bounded by system limits. It is the layer that allows handling unexpected cases, and applying ethical and practical judgment in context.

4. Version Management and References

المراجعة البشرية ليست عرقلة للعلم؛ إنها طريقة لجعل القرار قابلاً للمساءلة ومحدوداً بحدود النظام. هي الطبقة التي تسمح بالتعامل مع الحالات غير المتوقعة، وتطبيق الحكم الأخلاقي والعملي في السياق.

A good screening system needs precise version management. Reference databases are constantly changing: new sequences are added, old classifications are updated, and errors are corrected. If you do not document the version you used, you cannot

reconstruct the decision a year later. Therefore, you must record the version of each database you used, its update date, and any policy modifications.

4. إدارة الإصدارات والمراجع

نظام الفحص الجيد يحتاج إلى إدارة دقيقة للإصدارات. قواعد البيانات المرجعية تتغير باستمرار: تُضاف تسلسلات جديدة، وتُحدّث تصنيفات القديمة، وتُصحح الأخطاء. إذا لم توثق الإصدار الذي استخدمته، فلن يمكن إعادة بناء القرار بعد عام. لذلك، يجب تسجيل إصدار كل قاعدة بيانات استخدمتها، وتاريخ تحديثها، وأي تعديلات على السياسات.

ما يجب توثيقه:

- إصدار قاعدة البيانات المرجعية (مثل RefSeq 2025-01-15).

- تاريخ التحديث.

- السياسات المطبقة (مثل عتبة التشابه، قواعد التصنيف).

- أي استثناءات أو قرارات يدوية.

5. Practical Application: Educational Simulation of a Screening Pipeline

In the educational context, a sequence screening pipeline can be simulated using synthetic data, without accessing sensitive databases. The goal is to teach principles: input, screening, classification, review, and recording.

5. التطبيق العملي: محاكاة تعليمية لخط أنابيب فحص

في السياق التعليمي، يمكن محاكاة خط أنابيب فحص التسلسل باستخدام بيانات اصطناعية، دون الوصول إلى قواعد بيانات حساسة. الهدف هو تعليم المبادئ: الإدخال، الفحص، التصنيف، المراجعة، والتسجيل.

```
    # المرحلة 2: تصنيف المخاطر
    max_similarity = max(r["similarity"] for r in results) if results else 0
    if max_similarity >= similarity_threshold:
        risk = "high"
        recommended_action = "human_review_required"
    elif max_similarity >= 0.5:
        risk = "medium"
        recommended_action = "human_review_recommended"
    else:
        risk = "low"
        recommended_action = "auto_approve"
# محاكاة تعليمية لفحص التسلسل
def simulate_screening_pipeline(sequence, reference_db, similarity_threshold=0.8):
    """
```

محاكاة بسيطة لخط أنابيب فحص التسلسل.

للاستخدام التعليمي فقط (بيانات اصطناعية).

```
    """
    # المرحلة 1: الفحص الآلي (محاكاة)
    results = []
    for name, ref_seq in reference_db.items():
        matches = sum(1 for a, b in zip(sequence, ref_seq) if a == b)
        similarity = matches / len(sequence) if len(sequence) > 0 else 0
        results.append({"name": name, "similarity": similarity})
# المرحلة 3: تسجيل القرار (محاكاة)
    decision_record = {
        "sequence": sequence[:20] + "...",
        "max_similarity": max_similarity,
        "risk": risk,
        "action": recommended_action,
        "reviewer": "simulated_reviewer",
        "date": "2025-01-15",
        "database_version": "synthetic_v1.0"
    }
    return {
        "results": results,
        "decision": decision_record
    }

# بيانات اصطناعية
synthetic_db = {
    "toxin_A": "GACTAGCTAGCTAGCTAGCNGG",
    "toxin_B": "GACTAGCTAGCTAGCTATCNGG",
    "safe_seq": "ATGCGTACGTAGCTAGCTAGCTAGCTAGC"
}

query = "GACTAGCTAGCTAGCTAGCNGG"
output = simulate_screening_pipeline(query, synthetic_db)
print("نتائج الفحص (محاكاة تعليمية):")
for r in output["results"]:
    print(f"  {r['name']}: التشابه {r['similarity']:.2f}")
print("\nقرار الفحص:")
for key, value in output["decision"].items():
    print(f"  {key}: {value}")
```

الدوال المعرفة هنا (simulate_screening_pipeline) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational simulation illustrates the principle: automated screening generates candidates, then classifies risk, recommends an action (human review or auto-approval), and records the decision. However, it does not replace a real system with updated databases and real human review.

6. Measuring Error Instead of Relying on Apparent Accuracy

هذه المحاكاة التعليمية توضح المبدأ: الفحص الآلي يولد مرشحين، ثم يُصنف المخاطر، ويوصي بإجراء (مراجعة بشرية أو قبول تلقائي)، ويسجل القرار. لكنها لا تغني عن نظام حقيقي مع قواعد بيانات محدثة ومراجعة بشرية حقيقية.

In screening systems, it is easy to focus on "accuracy" (how often the system classifies correctly). But accuracy can be misleading if the data is unbalanced (e.g., most sequences are low-risk). More important is measuring errors: how often does the system classify a dangerous sequence as safe (false negative)? And how often does it classify a safe sequence as dangerous (false positive)? In security contexts, a false negative (missing a threat) is often more serious than a false positive (false alarm).

6. قياس الخطأ بدل الاكتفاء بالدقة الظاهرية

في أنظمة الفحص، من السهل التركيز على "الدقة" (كم مرة يصنف النظام بشكل صحيح). لكن الدقة قد تكون مضللة إذا كانت البيانات غير متوازنة (مثلاً، معظم التسلسلات منخفضة المخاطر). الأهم هو قياس الأخطاء: كم مرة يُصنف النظام تسلسل خطير على أنه آمن (سلبية كاذبة)؟ وكم مرة يُصنف تسلسل آمن على أنه خطير (إيجابية كاذبة)؟ في السياقات الأمنية، السلبية الكاذبة (تفويت تهديد) غالباً ما تكون أكثر خطورة من الإيجابية الكاذبة (تنبيه زائد).

مثال:

نظام يصنف 99% من التسلسلات بشكل صحيح (دقة عالية ظاهرياً).

لكنه يفوت 10% من التسلسلات الخطيرة (سلبيات كاذبة).

هذا النظام غير آمن، لأن الأخطاء الخطيرة غير مكتشفة.

Therefore, the screening system should be designed with a clear understanding of which types of errors are acceptable and which are not, and its performance should be documented on known test data.

7. Limits of Computational Screening

لذلك، يجب تصميم نظام الفحص مع فهم واضح لنوع الأخطاء المقبولة وغير المقبولة، وتوثيق أداء النظام على بيانات اختبار معروفة.

7. حدود الفحص الحاسوبي

فحص التسلسل الحاسوبي له حدود متأصلة:

· الاعتماد على المرجع: يعتمد على قواعد بيانات قد تكون غير مكتملة أو قديمة.

· التشابه ≠ النشاط: التشابه النصي لا يعادل النشاط البيولوجي.

Computational sequence screening has inherent limits:

· التسلسلات الجديدة: قد لا تكون التسلسلات الجديدة (غير المصنفة) موجودة في قواعد البيانات.

· Reference dependence: it relies on databases that may be incomplete or outdated.

· التطور السريع: مسببات الأمراض والسموم تتطور، وقد تظهر تسلسلات جديدة لا تشبه أي شيء معروف.

· Similarity ≠ activity: textual similarity does not equal biological activity.

· الهندسة العكسية: قد يحاول جهة خبيثة تصميم تسلسلات تتجاوز الفحص عن طريق تغيير طفيف في التسلسل مع الاحتفاظ بالوظيفة.

· New sequences: new (unclassified) sequences may not be in the databases.

· Rapid evolution: pathogens and toxins evolve, and new sequences may appear that do not resemble anything known.

· Reverse engineering: a malicious actor might try to design sequences that bypass screening by slightly changing the sequence while retaining function.

Therefore, computational screening should be part of a broader system that includes human review, regular updates, and international cooperation.

لذلك، يجب أن يكون الفحص الحاسوبي جزءاً من نظام أوسع يشمل المراجعة البشرية، وتحديثات منتظمة، وتعاوناً دولياً.

Chapter Summary

خلاصة الفصل

Sequence screening is a defensive governance pipeline, not just a search command. It combines automated screening, risk classification, human review, and decision recording. Automated screening alone is not sufficient; it needs human review to handle unexpected and ambiguous cases. Version management and decision documentation are the basis of auditability and reproducibility. Measuring errors (especially false negatives) is more important than focusing on apparent accuracy. Computational screening has inherent limits and must be part of a broader system. The next chapter will move to a deeper question: why sequence matching alone is not enough.

فحص التسلسل هو خط أنابيب حوكمة دفاعي، وليس مجرد أمر بحث. يجمع بين الفحص الآلي، وتصنيف المخاطر، والمراجعة البشرية، وتسجيل القرارات. الفحص الآلي ليس كافياً وحده؛ يحتاج إلى مراجعة بشرية للتعامل مع الحالات غير المتوقعة والملتبسة. إدارة الإصدارات وتوثيق القرارات هي أساس قابلية التدقيق وإعادة الإنتاج. قياس الأخطاء (خاصة السلبيات الكاذبة) أهم من التركيز على الدقة الظاهرية. الفحص الحاسوبي له حدود متأصلة، ويجب أن يكون جزءاً من نظام أوسع. الفصل القادم سينتقل إلى سؤال أعمق: لماذا لا تكفي مطابقة التسلسل وحدها؟

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Sequence Screening" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Sequence Screening" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "sequence screening can detect all dangerous sequences with 100% accuracy." This is incorrect; screening relies on databases that may be incomplete, and similarity does not equal activity. The correct statement is: "sequence screening is a defensive tool that generates candidates for review and classifies risk, but it cannot guarantee detection of all dangerous sequences, and requires human review and continuous updates." A simple test: Practical check: "Does the database the system relies on include all known dangerous sequences? Is it up to date?".

مثال على التجاوز: القول إن "فحص التسلسل يمكنه اكتشاف جميع التسلسلات الخطيرة بدقة 100%". هذا غير صحيح؛ الفحص يعتمد على قواعد بيانات قد تكون غير مكتملة، والتشابه لا يساوي النشاط. الصحيح هو: "فحص التسلسل هو أداة دفاعية تولد مرشحين للمراجعة، وتصنف المخاطر، لكنه لا يمكنه ضمان اكتشاف جميع التسلسلات الخطيرة، ويتطلب مراجعة بشرية وتحديثات مستمرة." اختبار بسيط: فحص تطبيقي: "هل قاعدة البيانات التي يعتمد عليها النظام تشمل جميع التسلسلات الخطيرة المعروفة؟ وهل هي محدثة؟"ـ

Example: A screening result showing 90% similarity to a known toxin sequence is a computational representation. Claiming that "this sequence is a dangerous toxin" is a biological claim not proven by similarity alone. Similarity prioritizes the sequence for review, but proving it is a toxin requires additional evidence (such as functional tests, or biological context). Practical test: before classifying a sequence as dangerous based on automated screening, verify the context and additional evidence.

مثال: نتيجة فحص تعطي تشابهاً 90% مع تسلسل سم معروف هي تمثيل حسابي. الادعاء بأن "هذا التسلسل هو سم خطير" هو ادعاء بيولوجي لا يثبته التشابه وحده. التشابه يرشح التسلسل للمراجعة، لكن إثبات أنه سم يتطلب أدلة

إضافية (مثل اختبارات وظيفية، أو سياق بيولوجي). اختبار عملي: قبل أن تصنف تسلسلاً على أنه خطير بناءً على الفحص الآلي، تحقق من السياق والأدلة الإضافية.

مثال تطبيقي على نظام فحص تسلسل:

· المدخلات: تسلسل استعلام، قاعدة بيانات مرجعية (مع الإصدار)، عتبة تشابه، سياسات تصنيف.

· الافتراضات: أن قاعدة البيانات تمثل المعرفة الحالية بالتسلسلات الخطيرة، وأن عتبة التشابه المختارة مناسبة، وأن السياسات تعكس القيم الأخلاقية والقانونية.

· المخرجات: تصنيف مخاطر، وتوصية بإجراء (مراجعة بشرية أو قبول تلقائي)، وسجل قرار.

· القيود: النظام لا يكتشف التسلسلات غير المصنفة، ولا يتنبأ بالنشاط البيولوجي بدقة، ويعتمد على تحديثات منتظمة.

Applied example of a sequence screening system:

اختبار: قبل الاعتماد على نظام فحص، تحقق من تاريخ تحديث قاعدة البيانات، وعتبة التشابه المستخدمة، وسياسات التصنيف المطبقة.

· Inputs: query sequence, reference database (with version), similarity threshold, classification policies.

· Assumptions: that the database represents current knowledge of dangerous sequences, that the chosen similarity threshold is appropriate, and that the policies reflect ethical and legal values.

· Outputs: risk classification, action recommendation (human review or auto-approval), and decision record.

· Limitations: the system does not detect unclassified sequences, nor accurately predict biological activity, and relies on regular updates.

Test: before relying on a screening system, check the database update date, the similarity threshold used, and the classification policies applied.

عند استخدام نظام فحص، يجب تسجيل ما يلي:

· المصدر: التسلسل المستهدف (مصدره، تاريخه، الغرض منه).

· الإصدار: إصدار قاعدة البيانات المرجعية، وإصدار النظام البرمجي.

· المعلمات: عتبة التشابه، سياسات التصنيف، أي استثناءات أو قرارات يدوية.

· التحويلات: أي خطوات لتنقية البيانات أو تحويلها قبل الفحص.

When using a screening system, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لمراجع بعد عام، هل يستطيع فهم كيفية اتخاذ القرار، وما هي السياسات والمراجع التي استُخدمت؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the target sequence (its source, date, purpose).

· Version: the reference database version, and the software system version.

· Parameters: the similarity threshold, classification policies, and any exceptions or manual decisions.

· Transformations: any steps to clean or transform data before screening.

Test: Practical check: "If I gave this record to a reviewer a year later, could they understand how the decision was made, and what policies and references were used?" If the answer is no, the documentation is insufficient.

قبل تطبيق نظام فحص على بيانات حقيقية، اختبره على بيانات اصطناعية ذات نتائج معروفة. أنشئ مجموعة من التسلسلات الاصطناعية التي تتضمن تسلسلات خطيرة وآمنة معروفة. شغّل نظام الفحص وتحقق من أنه يصنفها بشكل صحيح (حساسية ونوعية). اختبر أيضاً حالات حدية (مثل تسلسلات متشابهة ولكنها ليست خطيرة) لترى كيف يتعامل النظام مع الغموض. إذا نجح الاختبار، فأنت واثق من أن النظام يعمل كما هو متوقع، وأي حالة مع بيانات حقيقية تعكس تعقيدات العالم الحقيقي التي قد تتطلب مراجعة بشرية.

Before applying a screening system to real data, test it on synthetic data with known outcomes. Create a set of synthetic sequences including known dangerous and safe sequences. Run the screening system and verify that it classifies them correctly (sensitivity and specificity). Also test edge cases (such as similar but non-dangerous sequences) to see how the system handles ambiguity. If the test succeeds, the workflow is confident that the system works as expected, and any case with real data reflects real-world complexities that may require human review.

Chapter 33 — Why Sequence Matching Is Not Enough

الفصل 33 — لماذا لا تكفي مطابقة التسلسل؟

Deeper Conceptual Explanation

الفكرة المركزية: sequence similarity، الوظيفة، false negatives، function-aware screening.

شرح أعمق للمفهوم

A sequence can resemble a known element without sharing the same context or function. Databases are incomplete and changing, and algorithms use approximate

criteria. Assessment should therefore combine context, data quality, independent sources, and human review when consequences are important.

قد يكون التسلسل متشابهاً مع عنصر معروف من دون أن يعني ذلك السياق نفسه أو الوظيفة نفسها. كما أن قواعد البيانات ناقصة ومتغيرة، والخوارزميات تستخدم معايير تقريبية. لذلك يجب أن يجمع التقييم بين السياق، وجودة البيانات، ومصادر مستقلة، ومراجعة بشرية عندما تكون العواقب مهمة.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Similarity ≠ Activity

A common mistake is assuming that any sequence with high similarity to a known sequence will necessarily perform the same function. This is incorrect. Biological activity depends on multiple factors beyond textual similarity:

1. التشابه ≠ النشاط

من الأخطاء الشائعة افتراض أن أي موقع يحمل تشابهاً عالياً مع تسلسل معروف سيؤدي بالضرورة نفس الوظيفة. هذا غير صحيح. النشاط البيولوجي يعتمد على عوامل متعددة تتجاوز التشابه النصي:

· السياق الجينومي: موقع التسلسل في الجينوم، والعناصر التنظيمية المحيطة به، تؤثر على التعبير والوظيفة.

· Genomic context: the sequence's location in the genome, and surrounding regulatory elements, affect expression and function.

· البنية ثلاثية الأبعاد: البروتينات المتشابهة في التسلسل قد تطوى بشكل مختلف، مما يغير وظيفتها.

· Three-dimensional structure: proteins with similar sequences may fold differently, changing their function.

· التعديلات اللاحقة للترجمة: قد تُعدّل البروتينات كيميائياً بعد الترجمة، مما يغير نشاطها.

· Post-translational modifications: proteins may be chemically modified after translation, altering their activity.

· العوامل الخلوية: وجود بروتينات أخرى، والظروف الخلوية، تؤثر على النشاط الفعلي.

· Cellular factors: the presence of other proteins, and cellular conditions, affect actual activity.

مثال:

تسلسل X: تشابه 90% مع سم معروف، لكنه يفتقر إلى الموقع النشط ← لا يؤدي الوظيفة السامة.

تسلسل Y: تشابه 60% مع سم معروف، لكنه يحتفظ بالموقع النشط ← قد يؤدي الوظيفة السامة.

For this reason, similarity should be treated as an initial filter, not as a final judgment. Tools that rely only on similarity may produce false positives (classifying a safe sequence as dangerous) and false negatives (missing a dangerous sequence).

2. The New Challenge: AI and Decoupling Sequence from Function

لهذا السبب، يجب التعامل مع التشابه كمرشح أولي، وليس كحكم نهائي. الأدوات التي تعتمد فقط على التشابه قد تنتج إيجابيات كاذبة (تصنيف تسلسل آمن على أنه خطير) وسلبيات كاذبة (تفويت تسلسل خطير).

The rapid advancement of AI-enabled biodesign tools is changing the rules of the game. These tools can generate sequences that perform specific functions but bear no resemblance to any known sequence in nature or in databases. In other words, AI can "decouple" sequence from function: the same hazardous function can be achieved with sequences completely different from known ones.

2. التحدي الجديد: الذكاء الاصطناعي وفك الارتباط بين التسلسل والوظيفة

التقدم السريع في أدوات التصميم الحيوي المدعومة بالذكاء الاصطناعي يغير قواعد اللعبة. هذه الأدوات يمكنها توليد تسلسلات تؤدي وظائف محددة، لكنها لا تشبه أي تسلسل معروف في الطبيعة أو في قواعد البيانات. بعبارة أخرى، يمكن للذكاء الاصطناعي "فك الارتباط" بين التسلسل والوظيفة: نفس الوظيفة الخطيرة يمكن تحقيقها بتسلسلات مختلفة تماماً عن تلك المعروفة.

النهج التقليدي: البحث عن تسلسلات تشبه ما هو معروف.

النهج الجديد: البحث عن تسلسلات تؤدي وظائف خطيرة، بغض النظر عن شكلها.

This means that screening systems based solely on sequence matching may become ineffective against AI-designed threats. Sequences that these systems might miss (false negatives) represent a serious security gap.

3. Function-Aware Screening

هذا يعني أن أنظمة الفحص القائمة على مطابقة التسلسل وحدها قد تصبح غير فعالة ضد التهديدات المصممة بالذكاء الاصطناعي. التسلسلات التي قد تفوتها هذه الأنظمة (السلبيات الكاذبة) تمثل ثغرة أمنية خطيرة.

In response to these challenges, researchers are developing a new approach: function-based screening. Instead of asking "does this sequence resemble something known?", this approach asks "could this sequence perform a hazardous function?".

3. الفحص القائم على الوظيفة (Function-Aware Screening)

One example of this approach is SeqScreen, which uses machine learning to classify sequences based on their potential functions, not just on their similarity to known sequences. SeqScreen uses a curated set of "Functions of Sequences of Concern" (FunSoCs), which are specific functions associated with microbial pathogenesis.

استجابة لهذه التحديات، يعمل الباحثون على تطوير نهج جديد: الفحص القائم على الوظيفة. بدلاً من السؤال "هل يشبه هذا التسلسل شيئاً معروفاً؟"، يسأل هذا النهج "هل يمكن لهذا التسلسل أن يؤدي وظيفة خطيرة؟"ـ

أحد الأمثلة على هذا النهج هو أداة SeqScreen، التي تستخدم التعلم الآلي لتصنيف التسلسلات بناءً على وظائفها المحتملة، وليس فقط على تشابهها مع تسلسلات معروفة. تستخدم SeqScreen مجموعة من "وظائف التسلسلات محل الاهتمام" (FunSoCs) المنسقة يدوياً، وهي وظائف محددة مرتبطة بإمراضية الميكروباتـ.

مقارنة بين النهجين:

النهج القائم على التشابه: يبحث عن تطابق مع قاعدة بيانات.

النهج القائم على الوظيفة: يتنبأ بالوظيفة من التسلسل، بغض النظر عن التشابه.

Researchers from the National Institute of Standards and Technology (NIST) advocate for a hybrid architecture that combines sequence-based and function-based screening, where function-specific models complement existing screening systems. The focus starts with toxins as the most tractable targets, then expands to more complex pathogenic functions.

يدعو باحثون من المعهد الوطني للمعايير والتقنية (NIST) إلى بنية هجينة تجمع بين الفحص القائم على التسلسل والفحص القائم على الوظيفة، حيث تكمل النماذج الخاصة بالوظيفة أنظمة الفحص القائمة حالياً. يبدأ التركيز على السموم كأهداف أكثر قابلية للتعامل، ثم يتوسع ليشمل وظائف إمراضية أكثر تعقيداًـ

4. Open Challenges

The transition to function-based screening faces many challenges:

4. التحديات المفتوحة

الانتقال إلى الفحص القائم على الوظيفة يواجه تحديات عديدة:

· توحيد تعريفات "الوظائف محل الاهتمام": ما هي الوظائف التي تعتبر خطيرة؟ هذا سؤال أخلاقي وعلمي يحتاج إلى إجماع دولي.

· Standardizing definitions of "functions of concern": which functions are considered dangerous? This is an ethical and scientific question requiring international consensus.

· بناء بنية تحتية للتقييم: الحاجة إلى مجموعات بيانات مرجعية ومعايير لتقييم أدوات الفحص القائم على الوظيفة.

· Building evaluation infrastructure: the need for reference datasets and benchmarks to evaluate function-based screening tools.

· التوسع التجريبي: الحاجة إلى توصيف تجريبي للوظائف الخطيرة لتوليد بيانات تدريب للنماذج.

· Experimental expansion: the need for experimental characterization of hazardous functions to generate training data for models.

· السلبيات الكاذبة والإيجابيات الكاذبة: تحقيق التوازن بين تجنب تفويت التهديدات (السلبيات الكاذبة) وتجنب الإفراط في الإبلاغ (الإيجابيات الكاذبة).

· False negatives and false positives: balancing the avoidance of missing threats (false negatives) with avoiding over-reporting (false positives).

5. Implications for Practice

For the practitioner in bioinformatics, there are practical lessons from this chapter:

5. الآثار المترتبة على الممارسة العملية

بالنسبة للممارس في المعلوماتية الحيوية، هناك دروس عملية من هذا الفصل:

· لا تثق بالتشابه وحده: التشابه النصي قد يكون مؤشراً، لكنه ليس دليلاً. استخدم أدوات متعددة، واطلب مراجعة بشرية للحالات الملتبسة.

· Do not trust similarity alone: textual similarity may be an indicator, but it is not proof. Use multiple tools, and request human review for ambiguous cases.

· كن على دراية بالقيود: أنظمة الفحص القائمة على التشابه لها حدود متأصلة، خاصة مع التقدم في الذكاء الاصطناعي.

· Be aware of limitations: similarity-based screening systems have inherent limits, especially with advances in AI.

· تابع التطورات: الفحص القائم على الوظيفة هو مجال ناشئ، ومن المتوقع أن تتطور الأدوات والمعايير بسرعة.

· Follow developments: function-based screening is an emerging field, and tools and standards are expected to evolve rapidly.

· وثق القرارات: كما هو الحال مع أي نظام أمني، توثيق القرارات والأساس المنطقي لها هو أساس قابلية التدقيق والمراجعة.

· Document decisions: as with any security system, documenting decisions and their rationale is the basis of auditability and review.

Chapter Summary

خلاصة الفصل

Sequence matching alone is not sufficient for screening dangerous sequences. Textual similarity does not equal biological activity, and AI tools can generate dangerous sequences that resemble nothing known. Therefore, function-based screening is emerging as a complementary approach, aiming to detect sequences capable of hazardous functions regardless of their similarity. This approach faces challenges, but represents a necessary direction to keep pace with advances in biodesign. The next chapter will move to Gene Drive Constraints as a software governance tool.

مطابقة التسلسل وحدها ليست كافية لفحص التسلسلات الخطيرة. التشابه النصي لا يساوي النشاط البيولوجي، وأدوات الذكاء الاصطناعي يمكنها توليد تسلسلات خطيرة لا تشبه أي شيء معروف. لذلك، يبرز الفحص القائم على الوظيفة كنهج مكمل، يهدف إلى اكتشاف التسلسلات القادرة على أداء وظائف خطيرة بغض النظر عن تشابهها. هذا النهج يواجه تحديات، لكنه يمثل اتجاهاً ضرورياً لمواكبة التطورات في التصميم الحيوي. الفصل القادم سينتقل إلى قيود المحركات الجينية (Gene Drive Constraints) كأداة حوكمة برمجية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Why Sequence Matching Is Not Enough" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "لماذا لا تكفي مطابقة التسلسل؟" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "function-based screening can detect all dangerous sequences with 100% accuracy." This is incorrect; function-based screening is a promising approach, but it is still in its early stages and has its challenges. The correct statement is: "similarity-based screening has inherent limits, especially with advances in AI, and function-based screening represents a promising complementary approach, but it needs development and standardization." A simple test: Practical check: "Does this tool detect sequences that resemble nothing known?".

مثال على التجاوز: القول إن "الفحص القائم على الوظيفة يمكنه اكتشاف جميع التسلسلات الخطيرة بدقة 100%". هذا غير صحيح؛ الفحص القائم على الوظيفة هو نهج واعد، لكنه لا يزال في مراحله الأولى، وله تحدياته. الصحيح هو: "الفحص القائم على التشابه له حدود متأصلة، خاصة مع تقدم الذكاء الاصطناعي، والفحص القائم على الوظيفة يمثل

نهجاً مكملاً واعداً، لكنه يحتاج إلى تطوير وتوحيد." اختبار بسيط: فحص تطبيقي: "هل هذه الأداة تكتشف التسلسلات التي لا تشبه أي شيء معروف؟"ـ

Example: A screening result showing 90% similarity to a known toxin sequence, or predicting a toxic function, is a computational representation. Claiming that "this sequence is a dangerous toxin" is a biological claim not proven by similarity or prediction alone. Similarity and prediction prioritize the sequence for review, but proving it is a toxin requires additional evidence (such as functional tests, or biological context). Practical test: before classifying a sequence as dangerous based on automated screening, verify the context and additional evidence.

مثال: نتيجة فحص تعطي تشابهاً 90% مع تسلسل سم معروف، أو تتنبأ بوظيفة سامة، هي تمثيل حسابي. الادعاء بأن "هذا التسلسل هو سم خطير" هو ادعاء بيولوجي لا يثبته التشابه أو التنبؤ وحده. التشابه والتنبؤ يرشحان التسلسل للمراجعة، لكن إثبات أنه سم يتطلب أدلة إضافية (مثل اختبارات وظيفية، أو سياق بيولوجي). اختبار عملي: قبل أن تصنف تسلسلاً على أنه خطير بناءً على الفحص الآلي، تحقق من السياق والأدلة الإضافية.

مثال تطبيقي على الفحص القائم على الوظيفة:

· المدخلات: تسلسل استعلام، نموذج تعلم آلي مدرب على وظائف معروفة، وقاعدة بيانات مرجعية للوظائف.

· الافتراضات: أن النموذج يعكس العلاقة بين التسلسل والوظيفة بدقة معقولة، وأن الوظائف المصنفة على أنها "محل اهتمام" محددة بشكل واضح.

· المخرجات: تنبؤ بالوظيفة المحتملة، وتصنيف مخاطر، وتوصية بإجراء.

· القيود: النماذج الحالية قد لا تغطي جميع الوظائف الخطيرة، وقد تنتج إيجابيات كاذبة أو سلبيات كاذبة. تعتمد جودة التنبؤ على جودة بيانات التدريب.

Applied example of function-based screening:

اختبار: قبل الاعتماد على أداة فحص قائمة على الوظيفة، تحقق من دقة النموذج على بيانات اختبار معروفة، وافهم حدودها.

· Inputs: query sequence, a machine learning model trained on known functions, and a reference database of functions.

· Assumptions: that the model reflects the relationship between sequence and function with reasonable accuracy, and that the functions classified as "of concern" are clearly defined.

· Outputs: a prediction of potential function, risk classification, and action recommendation.

· Limitations: current models may not cover all dangerous functions, and may produce false positives or false negatives. Prediction quality depends on training data quality.

Test: before relying on a function-based screening tool, verify the model's accuracy on known test data, and understand its limitations.

عند استخدام أدوات الفحص (سواء القائمة على التشابه أو الوظيفة)، يجب تسجيل ما يلي:

· الإصدار: إصدار قاعدة البيانات المرجعية، وإصدار النموذج أو الأداة البرمجية.

· المعلمات: عتبة التشابه، أو معلمات النموذج، وأي سياسات تصنيف مطبقة.

When using screening tools (whether similarity-based or function-based), record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لمراجع بعد عام، هل يستطيع فهم كيفية اتخاذ القرار، وما هي النماذج والمراجع التي استُخدمت؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Version: the reference database version, and the model or software version.

· Parameters: the similarity threshold, or model parameters, and any classification policies applied.

Test: Practical check: "If I gave this record to a reviewer a year later, could they understand how the decision was made, and what models and references were used?" If the answer is no, the documentation is insufficient.

قبل تطبيق أي أداة فحص (تقليدية أو قائمة على الوظيفة) على بيانات حقيقية، اختبرها على بيانات اصطناعية ذات نتائج معروفة. أنشئ مجموعة من التسلسلات الاصطناعية التي تتضمن تسلسلات خطيرة وآمنة معروفة، بالإضافة إلى تسلسلات مشابهة ولكنها غير خطيرة. شغّل أداة الفحص وتحقق من أنها تصنفها بشكل صحيح (حساسية ونوعية). إذا نجح الاختبار، فأنت واثق من أن الأداة تعمل كما هو متوقع، وأي حالة مع بيانات حقيقية تعكس تعقيدات العالم الحقيقي التي قد تتطلب مراجعة بشرية.

Before applying any screening tool (traditional or function-based) to real data, test it on synthetic data with known outcomes. Create a set of synthetic sequences including known dangerous and safe sequences, as well as similar but non-dangerous sequences. Run the screening tool and verify that it classifies them correctly (sensitivity and specificity). If the test succeeds, the workflow is confident that the tool works as expected, and any case with real data reflects real-world complexities that may require human review.

Part 10 — The biobrick_security Educational Package

الباب 10 — مشروع biobrick_security كحزمة تعليمية

Chapter 34 — Gene Drive Constraints

الفصل

الفكرة المركزية: القيود البرمجية، fail-closed، الصلاحيات، التوقف، المراجعة.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

يُستخدم مفهوم gene drive هنا لتوضيح كيف يمكن تحويل مفهوم بيولوجي حساس إلى قواعد حوكمة وفحص دفاعي. الهدف ليس بناء نظام gene drive، بل فهم لماذا تحتاج الأنظمة ذات الانتشار المحتمل إلى ضوابط توقف، ومراجعة، وتوثيق، وحدود استخدام واضحة.

Gene drive is used here to show how a sensitive biological concept can be translated into governance and defensive checks. The goal is not to build a gene-drive system, but to understand why systems with potential propagation require stopping controls, review, documentation, and clear use boundaries.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Are Gene Drives؟

1. ما هي المحركات الجينية؟

A gene drive is a genetic system that increases the probability of a specific trait being passed from one generation to the next, bypassing Mendelian laws which state that each trait is inherited at a 50% rate (in diploid organisms). In a gene drive, the trait is designed to be inherited at nearly 100%, allowing it to spread rapidly through populations. This technology relies on systems like CRISPR-Cas9 to insert a genetic modification into the genome, with the guide designed to cut the genome at a specific site, then using the cell's repair mechanism to copy the modification to the homologous chromosome.

المحرك الجيني هو نظام وراثي يزيد من احتمالية انتقال صفة معينة من جيل إلى آخر، متجاوزاً قوانين المندل التي تنص على أن كل صفة تنتقل بنسبة 50% (في الكائنات ثنائية الصيغة الصبغية). في المحرك الجيني، تُصمم الصفة بحيث تنتقل بنسبة تصل إلى 100% تقريباً، مما يسمح لها بالانتشار بسرعة عبر السكان. هذه التقنية تعتمد على

أنظمة مثل CRISPR-Cas9 لتضمين تعديل جيني في الجينوم، مع تصميم الدليل بحيث يقطع الجينوم في موقع محدد، ثم يستخدم آلية إصلاح الخلية لنسخ التعديل إلى الكروموسوم المتماثل.

مثال:

في نظام طبيعي، صفة معينة تنتقل من أب إلى ابن بنسبة 50%.

في محرك جيني، نفس الصفة تنتقل بنسبة تصل إلى 100%، مما يسمح لها بالانتشار عبر السكان بسرعة.

Gene drives hold tremendous promise in areas such as vector-borne disease control (e.g., modifying mosquitoes to resist malaria), eradicating invasive species, and improving agriculture. But they also carry significant risks: unintended spread, unforeseen environmental effects, and potential for misuse.

2. Software Constraints as a Governance Model

المحركات الجينية تحمل وعوداً هائلة في مجالات مثل مكافحة الأمراض المنقولة بالنواقل (مثل تعديل البعوض لمقاومة الملاريا)، والقضاء على الأنواع الغازية، وتحسين الزراعة. لكنها تحمل أيضاً مخاطر كبيرة: انتشار غير مقصود، تأثيرات بيئية غير متوقعة، وإمكانية إساءة الاستخدام.

Instead of teaching how to build a gene drive, this chapter presents software constraints as a general governance model for critical systems. The basic idea is that the software path does not proceed to a sensitive step if documented approval or condition verification is absent. This model is not limited to gene drives; it applies to any software system that could have high consequences.

2. القيود البرمجية كنموذج حوكمة

بدلاً من تعليم كيفية بناء محرك جيني، يعرض هذا الفصل القيود البرمجية كنموذج عام للحوكمة في الأنظمة الحرجة. الفكرة الأساسية هي أن المسار البرمجي لا ينتقل إلى خطوة حساسة إذا غابت موافقة موثقة أو تحقق من الشروط. هذا النموذج لا يقتصر على المحركات الجينية؛ بل ينطبق على أي نظام برمجي يمكن أن يكون له عواقب عالية.

مبدأ عام: لا تنتقل إلى الخطوة التالية إلا إذا استوفيت جميع الشروط المطلوبة.

هذا المبدأ يُعرف بـ "الفشل مغلق" (fail-closed).

In traditional systems, the default behavior may be to continue if nothing prevents it (assume allow). In critical systems, the default should be to stop (assume deny). This is the essence of the fail-closed principle.

في الأنظمة التقليدية، قد يكون السلوك الافتراضي هو الاستمرار إذا لم يمنع شيء (افتراض السماح). في الأنظمة الحرجة، يجب أن يكون السلوك الافتراضي هو التوقف (افتراض المنع). هذا هو جوهر مبدأ fail-closed.

3. Authorization and Stop Conditions

3. الصلاحيات (Authorization) وشروط التوقف (Stop Conditions)

Authorization is not just a button or a single permission. In the proposed model, sensitive steps require a set of preconditions:

الصلاحية ليست مجرد زر أو إذن منفرد. في النموذج المقترح، تتطلب الخطوات الحساسة مجموعة من الشروط المسبقة:

· إذن صريح: موافقة من شخص أو جهة مخولة، موثقة ومسجلة.

· Explicit authorization: approval from an authorized person or body, documented and recorded.

· تحقق من الشروط: التأكد من أن جميع الشروط المطلوبة للخطوة قد استُوفيت (مثل اكتمال الفحوصات الأمنية، ومراجعة الأخلاقيات).

· Condition verification: ensuring that all required conditions for the step have been met (e.g., completion of security checks, ethics review).

· سجل قرار: تسجيل كل قرار، مع الأساس المنطقي له، وتاريخه، وهوية المسؤول.

· Decision record: recording every decision, with its rationale, date, and responsible party.

· قابلية التدقيق: يجب أن يكون من الممكن تتبع كل قرار إلى مصدره، ومراجعته لاحقاً.

· Auditability: it should be possible to trace every decision back to its source, and review it later.

```
# مثال تعليمي لقيد برمجي (محاكاة)
def perform_sensitive_step(data, authorization, conditions_met, audit_log):
    if not authorization:
        audit_log.append("فشل: لا يوجد إذن صريح")
        return "مرفوض"
    if not conditions_met:
        audit_log.append("فشل: الشروط غير مستوفاة")
```

return "معلق - مراجعة مطلوبة"

تنفيذ الخطوة الحساسة

audit_log.append("نجاح: تم تنفيذ الخطوة بعد التحقق من الشروط")

```
    return "تم التنفيذ"
```

الدوال المعرفة هنا (perform_sensitive_step) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة.

القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational example shows that the sensitive step is only executed after verifying authorization and conditions, with the decision recorded in the audit log.

4. Applying Fail-Safe Principles

هذا المثال التعليمي يوضح أن الخطوة الحساسة لا تُنفذ إلا بعد التحقق من الصلاحية والشروط، مع تسجيل القرار في سجل التدقيق.

The fail-closed principle is part of a broader set of fail-safe principles. In critical systems, design should ensure that failure (e.g., software error, loss of connectivity, or cyberattack) does not lead to an unsafe state. Instead, the system should transition to a known safe state (such as stopping, or escalating to human review).

4. تطبيق مبادئ الفشل الآمن (Fail-Safe Principles)

مبدأ fail-closed هو جزء من مجموعة أوسع من مبادئ الفشل الآمن. في الأنظمة الحرجة، التصميم يجب أن يضمن أن الفشل (مثل خطأ برمجي، أو فقدان الاتصال، أو هجوم سيبراني) لا يؤدي إلى حالة غير آمنة. بدلاً من ذلك، يجب أن ينتقل النظام إلى حالة آمنة معروفة (مثل التوقف، أو التصعيد للمراجعة البشرية).

مبادئ الفشل الآمن:

1. الفشل مغلق (fail-closed): إذا فشل النظام، قف.

2. الفشل بأمان (fail-safe): إذا فشل النظام، انتقل إلى حالة آمنة معروفة.

3. الفشل معزول (fail-isolated): إذا فشل جزء من النظام، لا يؤثر ذلك على الأجزاء الأخرى.

4. الفشل مع المراجعة (fail-review): إذا فشل النظام، قم بتصعيد الحالة للمراجعة البشرية.

In the context of gene drives, this means the system should be designed so that it cannot be operated without explicit authorization, and that any error leads to stopping or review, not to unauthorized execution.

5. Audit Logs and Explicit Approval

في سياق المحركات الجينية، هذا يعني أن النظام يجب أن يكون مصمماً بحيث لا يمكن تشغيله بدون إذن صريح، وأن أي خطأ يؤدي إلى التوقف أو المراجعة، وليس إلى تنفيذ غير مصرح به.

Audit logs are the backbone of accountability in critical systems. Every important action should be recorded, including:

5. سجلات التدقيق والموافقة الصريحة

سجل التدقيق هو العمود الفقري للمساءلة في الأنظمة الحرجة. يجب أن يسجل كل إجراء مهم، بما في ذلك:

· من طلب الإجراء.

· متى طُلب.

· ما هو الإجراء المطلوب.

· ما هي الشروط التي تحققت.

· Who requested the action.

· ما هو القرار النهائي.

· When it was requested.

· من وافق (إذا كانت هناك موافقة).

· What action was requested.

· الأساس المنطقي للقرار.

· What conditions were met.

· What was the final decision.

· Who approved (if approval was given).

· The rationale for the decision.

مثال لسجل تدقيق:

2025-01-15 14:30:12 | المستخدم: researcher_001 | الإجراء: تعديل جين X | الشروط: تم التحقق من 3 شروط من أصل 4 | القرار: معلق - مراجعة مطلوبة | المراجع: committee_lead

2025-01-15 16:45:33 | المستخدم: committee_lead | الإجراء: مراجعة طلب تعديل جين X | القرار: مرفوض - عدم كفاية الأدلة على السلامة

A good audit log allows tracking decisions, identifying errors, improving processes, and providing accountability.

6. Practical Application: Simulating a Constraint System

سجل التدقيق الجيد يسمح بتتبع القرارات، وتحديد الأخطاء، وتحسين العمليات، وتوفير المساءلة.

In the educational context, a gene drive constraint system can be simulated using synthetic data and hypothetical approvals. The goal is to teach principles, not to implement a real system.

6. التطبيق العملي: محاكاة نظام قيود

في السياق التعليمي، يمكن محاكاة نظام قيود المحرك الجيني باستخدام بيانات اصطناعية وموافقات وهمية. الهدف

هو تعليم المبادئ، وليس تنفيذ نظام حقيقي.

```
# محاكاة تعليمية لنظام قيود المحرك الجيني
def gene_drive_constraint_simulation(request, approval_required=True, conditions_met=True,
audit_log=None):
    if audit_log is None:
        audit_log = []
# الخطوة 1: تسجيل الطلب
    audit_log.append(f"طلب: {request}")
# الخطوة 2: التحقق من الصلاحية
    if approval_required and not request.get("approved", False):
        audit_log.append("فشل: لا يوجد موافقة صريحة")
        return {"status": "denied", "reason": "no explicit approval", "audit": audit_log}
# الخطوة 3: التحقق من الشروط
    if not conditions_met:
        audit_log.append("فشل: الشروط غير مستوفاة")
return {"status": "pending_review", "reason": "conditions not met",
"audit": audit_log}
# مثال على الاستخدام
request = {"action": "gene_drive_release", "approved": True, "target":
"mosquito_population"}
result = gene_drive_constraint_simulation(request, approval_required=True,
conditions_met=True)
# الخطوة 4: تنفيذ الإجراء (محاكاة)
audit_log.append("نجاح: تم تنفيذ الإجراء بعد التحقق من جميع الشروط")
    return {"status": "executed", "audit": audit_log}
print("نتائج المحاكاة التعليمية:")
for entry in result["audit"]:
    print(f"  {entry}")
print(f"الحالة النهائية: {result['status']}")
```

الدوال المعرفة هنا (gene_drive_constraint_simulation) تمثل حدودًا واضحة بين المدخلات

والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل

إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational simulation illustrates the principle: every sensitive step requires verification of authorization and conditions, and every decision is recorded.

However, it does not replace a real system with actual security policies and human review.

7. Limits of Software Constraints

هذه المحاكاة التعليمية توضح المبدأ: كل خطوة حساسة تتطلب تحققاً من الصلاحية والشروط، ويُسجل كل قرار. لكنها لا تغني عن نظام حقيقي مع سياسات أمنية ومراجعة بشرية فعلية.

Software constraints, no matter how strong, are not sufficient alone. There are limits that must be clear:

7. حدود القيود البرمجية

القيود البرمجية، مهما كانت قوية، ليست كافية وحدها. هناك حدود يجب أن تكون واضحة:

· الأخطاء البرمجية: قد تحتوي الأنظمة على أخطاء تسمح بتجاوز القيود.

· Software errors: systems may contain errors that allow bypassing constraints.

· الهجمات السيبرانية: قد يحاول جهة خبيثة اختراق النظام وتجاوز القيود.

· Cyberattacks: a malicious party may attempt to breach the system and bypass constraints.

· التواطؤ البشري: قد يتعاون أشخاص داخل المؤسسة لتجاوز القيود.

· Human collusion: individuals within the organization may collaborate to bypass constraints.

· الاعتماد على السياسات: القيود البرمجية تنفذ السياسات الموضوعة، لكنها لا تحل محل الحوكمة الأخلاقية والقانونية.

· Policy dependence: software constraints implement set policies, but they do not replace ethical and legal governance.

Therefore, software constraints should be part of a broader governance system that includes policies, human review, auditing, and international cooperation.

لذلك، يجب أن تكون القيود البرمجية جزءاً من نظام حوكمة أوسع يشمل السياسات، والمراجعة البشرية، والتدقيق، والتعاون الدولي.

Chapter Summary

خلاصة الفصل

Gene drives are a powerful technology that can edit genomes on a large scale, carrying significant promises and risks. Instead of teaching how to build gene

drives, this chapter presents software constraints as a general governance model for critical systems. The basic principle is that the software path does not proceed to a sensitive step if documented approval or condition verification is absent (the fail-closed principle). Sensitive steps require explicit authorization, condition verification, decision recording, and auditability. Audit logs are the backbone of accountability. Software constraints alone are not sufficient; they must be part of a broader governance system. The next chapter will move to DNA Watermarking as a tool for provenance tracking.

المحركات الجينية هي تقنية قوية يمكنها تعديل الجينومات على نطاق واسع، وتحمل وعوداً ومخاطر كبيرة. بدلاً من تعليم بناء المحركات الجينية، يعرض هذا الفصل القيود البرمجية كنموذج عام للحوكمة في الأنظمة الحرجة. المبدأ الأساسي هو أن المسار البرمجي لا ينتقل إلى خطوة حساسة إذا غابت موافقة موثقة أو تحقق من الشروط (مبدأ fail-closed). تتطلب الخطوات الحساسة إذناً صريحاً، وتحققاً من الشروط، وسجلاً للقرار، وقابلية للتدقيق. سجلات التدقيق هي العمود الفقري للمساءلة. القيود البرمجية وحدها ليست كافية؛ يجب أن تكون جزءاً من نظام حوكمة أوسع. الفصل القادم سينتقل إلى العلامات المائية للـ DNA (DNA Watermarking) كأداة لتتبع المنشأ.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Gene Drive Constraints" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Gene Drive Constraints" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "software constraints guarantee that gene drives will not be misused." This is incorrect; software constraints are important but not sufficient, and can be bypassed by software errors, attacks, or collusion. The correct statement is: "software constraints are part of a broader governance system aimed at preventing irresponsible use of gene drives, but they are not an absolute guarantee, and require human review and continuous auditing." A simple test: Practical check: "Is this software system immune to all types of attacks and human errors?".

مثال على التجاوز: القول إن "القيود البرمجية تضمن عدم إساءة استخدام المحركات الجينية". هذا غير صحيح؛ القيود البرمجية مهمة لكنها ليست كافية، وقد تُتجاوز بأخطاء برمجية أو هجمات أو تواطؤ. الصحيح هو: "القيود البرمجية هي جزء من نظام حوكمة أوسع يهدف إلى منع الاستخدام غير المسؤول للمحركات الجينية، لكنها ليست ضماناً مطلقاً، وتتطلب مراجعة بشرية وتدقيقاً مستمراً." اختبار بسيط: فحص تطبيقي: "هل هذا النظام البرمجي محصن ضد جميع أنواع الهجمات والأخطاء البشرية؟".

Example: A software system implementing constraints on gene drive execution is a computational representation of policies. Claiming that "this system guarantees biosafety" is a biological claim not proven by the system alone. The system can apply specific rules, but it cannot evaluate actual biological risks or ethical context. Practical test: before considering a software system "safe," verify the existence of human review and comprehensive policies.

مثال: نظام برمجي يطبق قيوداً على تنفيذ محرك جيني هو تمثيل حسابي للسياسات. الادعاء بأن "هذا النظام يضمن السلامة البيولوجية" هو ادعاء بيولوجي لا يثبته النظام وحده. النظام يمكنه تطبيق قواعد محددة، لكنه لا يستطيع تقييم المخاطر البيولوجية الفعلية أو السياق الأخلاقي. اختبار عملي: قبل أن تعتبر نظاماً برمجياً "آمناً"، تحقق من وجود مراجعة بشرية وسياسات شاملة.

مثال تطبيقي على نظام قيود محرك جيني:

· المدخلات: طلب (نوع الإجراء، الهدف، المعرف)، موافقات، شروط، سجل تدقيق.

· الافتراضات: أن الموافقات صحيحة وغير مزورة، وأن الشروط المحددة تعكس المتطلبات الأمنية والأخلاقية، وأن سجل التدقيق غير قابل للتلاعب.

· المخرجات: قرار (تنفيذ، تعليق، رفض)، وسجل تدقيق محدث.

· القيود: النظام لا يتحقق من صحة الموافقات خارج النظام، ولا يكتشف التواطؤ البشري، ولا يستبدل المراجعة الأخلاقية المستقلة.

Applied example of a gene drive constraint system:

اختبار: قبل الاعتماد على نظام قيود، تحقق من آليات المصادقة، وحماية سجل التدقيق، ووجود مراجعة بشرية للحالات المستثناة.

· Inputs: request (action type, target, identifier), approvals, conditions, audit log.

· Assumptions: that approvals are valid and not forged, that the specified conditions reflect security and ethical requirements, and that the audit log is tamper-proof.

· Outputs: decision (execute, hold, deny), and updated audit log.

· Limitations: the system does not verify the validity of approvals outside the system, nor detect human collusion, nor replace independent ethical review.

Test: before relying on a constraint system, verify authentication mechanisms, audit log protection, and the existence of human review for exceptional cases.

عند تصميم أو استخدام نظام قيود، يجب تسجيل ما يلي:

· المعلمات: أي معاملات تؤثر على قرارات النظام (مثل شروط التوقف، مستويات الصلاحية).

When designing or using a constraint system, record the following:

· Parameters: any parameters affecting system decisions (e.g., stop conditions, authorization levels).

قبل تطبيق أي نظام قيود على بيئة حقيقية، اختبره على سيناريوهات اصطناعية. أنشئ حالات اختبار تمثل مواقف مختلفة: طلب صحيح مع موافقات، طلب بدون موافقة، طلب مع شروط غير مستوفاة، محاولة تجاوز. تحقق من أن النظام يتعامل مع كل حالة بشكل صحيح وفقاً للسياسات المحددة، وأن سجل التدقيق يسجل كل قرار. إذا نجح الاختبار، فأنت واثق من أن النظام يعمل كما هو متوقع، وأي حالة مع بيئة حقيقية تعكس تعقيدات العالم الحقيقي التي قد تتطلب مراجعة بشرية.

Before applying any constraint system to a real environment, test it on synthetic scenarios. Create test cases representing different situations: a valid request with approvals, a request without approval, a request with unmet conditions, and an attempt to bypass. Verify that the system handles each case correctly according to the specified policies, and that the audit log records every decision. If the test succeeds, the workflow is confident that the system works as expected, and any case with a real environment reflects real-world complexities that may require human review.

Chapter 35 — DNA Watermarking

الفصل

الفكرة المركزية: التتبع، provenance، العلامات غير التشغيلية، قابلية الإسناد، حدودها.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

العلامة المائية في DNA مثال على تشفير أو ترميز معلومات داخل تسلسل اصطناعي مع الحفاظ على قيود محددة. قيمتها التعليمية في فصل طبقات البيانات عن المعنى البيولوجي: يمكن لتسلسل أن يحمل نمطاً مقصوداً، لكن اكتشاف النمط لا يثبت وظيفة حيوية.

DNA watermarking illustrates encoding information in a synthetic sequence while satisfying defined constraints. Its educational value is in separating data structure from biological meaning: a sequence can carry an intentional pattern, but detecting that pattern does not prove a biological function.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is DNA Watermarking؟

1. ما هي العلامة المائية للـ DNA؟

DNA watermarking is a technique that allows embedding identifying information (such as author identity, date, version number, or a unique identifier) within a DNA sequence, such that it does not affect the biological function of the sequence. This can be likened to a watermark on paper: a pattern invisible to the naked eye, but which proves the document's authenticity and origin. In the context of synthetic biology, watermarks are used to confirm sequence identity, track their use, and protect intellectual property.

العلامة المائية للـ DNA هي تقنية تسمح بتضمين معلومات تعريفية (مثل هوية المؤلف، التاريخ، رقم الإصدار، أو معرف فريد) داخل تسلسل DNA، بحيث لا تؤثر على الوظيفة البيولوجية للتسلسل. يمكن تشبيه ذلك بالعلامة المائية على الورق: نمط غير مرئي للعين المجردة، لكنه يثبت أصالة الوثيقة ومصدرها. في سياق البيولوجيا التركيبية، تُستخدم العلامات المائية لتأكيد هوية التسلسلات، وتتبع استخدامها، وحماية الملكية الفكرية.

Watermarks are typically created by small changes in the sequence that do not change the resulting protein (if any), such as synonymous codon substitutions. Advanced techniques also use encryption and error correction to ensure the watermark remains readable even in the event of mutations or synthesis errors.

تُصنع العلامات المائية عادةً بتغييرات صغيرة في التسلسل لا تغير البروتين الناتج (إن وُجد)، مثل استبدال الكودونات بما يعادلها (synonymous codon substitutions). تستخدم تقنيات متقدمة أيضاً تشفيراً وتصحيحاً للأخطاء لضمان بقاء العلامة المائية قابلة للقراءة حتى في حالة حدوث طفرات أو أخطاء في التوليف.

> مثال مبسط:
>
> التسلسل الأصلي: ATG GCT AGC (يترجم إلى M-A-S)

التسلسل مع علامة مائية: ATG GCC AGC (يترجم إلى M-A-S أيضاً، لكن الكودون الثاني مختلف)

2. Uses of DNA Watermarking

DNA watermarks have multiple uses in synthetic biology and biosecurity:

2. استخدامات العلامات المائية للـ DNA

Attribution and Intellectual Property Protection: With the increasing ease of designing and synthesizing DNA sequences, the need to prove authorship and protect intellectual property has become critical. Watermarks can prove that a given sequence was created in a specific laboratory, by a specific researcher, on a specific date. This is important in the growing bioeconomy, where synthetic sequences are used in pharmaceuticals, gene therapies, and biofuels.

استخدامات متعددة في البيولوجيا التركيبية والأمن الحيوي: للعلامات المائية للـ DNA

Traceability and Supply Chain Management: Watermarks can be used to track sequences across the supply chain: from the design lab, to the synthesis company, to the final lab. This allows detection of unauthorized use or leaks.

إثبات التأليف وحماية الملكية الفكرية: مع تزايد سهولة تصميم وتسليح تسلسلات DNA، أصبحت الحاجة إلى إثبات التأليف وحماية الملكية الفكرية أمراً بالغ الأهمية. يمكن للعلامات المائية أن تثبت أن تسلسلاً معيناً صُنع في مختبر معين، أو من قبل باحث معين، في تاريخ معين. هذا مهم في الاقتصاد الحيوي المتنامي، حيث تُستخدم التسلسلات الاصطناعية في الأدوية والعلاجات الجينية والوقود الحيوي.

Biosecurity: Watermarks can help identify sequences that may be dangerous, and trace their origin in case of misuse. Some techniques allow the creation of tamper-resistant genomic "fingerprints," making it difficult to modify the sequence without leaving a trace.

التتبع وإدارة سلسلة التوريد: يمكن استخدام العلامات المائية لتتبع التسلسلات عبر سلسلة التوريد: من مختبر التصميم، إلى شركة التوليف، إلى المختبر النهائي. هذا يسمح باكتشاف الاستخدام غير المصرح به أو التسريبات.

الأمن الحيوي: يمكن للعلامات المائية أن تساعد في التعرف على التسلسلات التي قد تكون خطرة، وتتبع مصدرها في حال إساءة الاستخدام. بعض التقنيات تسمح بإنشاء "بصمات" جينومية مقاومة للتلاعب، تجعل من الصعب تعديل التسلسل دون ترك أثر.

مثال على التتبع:

مختبر أ يطلب تسلسلاً من شركة توليف.

الشركة تضيف علامة مائية فريدة.

المختبر ب يحصل على التسلسل من مختبر أ.

إذا ظهر التسلسل في مكان غير مصرح به، يمكن تتبعه إلى مختبر أ عبر العلامة المائية.

3. Watermark Embedding Techniques

There are several ways to embed watermarks in DNA sequences, differing in complexity and strength:

3. تقنيات تضمين العلامات المائية

Synonymous Codon Substitution: This technique uses the fact that the genetic code is degenerate—multiple codons code for the same amino acid. By changing the codon to another that codes for the same amino acid, information can be embedded without changing the resulting protein. The DNAMark technique uses this approach to preserve protein function.

هناك عدة طرق لتضمين العلامات المائية في تسلسلات DNA، تختلف في تعقيدها وقوتها:

Non-Coding Region Embedding: Watermarks can be embedded in non-coding regions of the genome (such as introns or intergenic regions), where changes do not affect the protein. This approach provides more space for embedding, but may not be available in all sequences.

استبدال الكودونات المترادفة (Synonymous Codon Substitution): تستخدم هذه التقنية حقيقة أن الشفرة الوراثية متدهورة، أي أن عدة كودونات ترمز لنفس الحمض الأميني. بتغيير الكودون إلى آخر يرمز لنفس الحمض الأميني، يمكن تضمين معلومات دون تغيير البروتين الناتج. تقنية DNAMark تستخدم هذا الأسلوب للحفاظ على وظيفة البروتين.

Encryption and Digital Signature: The watermark can be encrypted using private keys, and a digital signature can be added to verify that the sequence has not been modified. This provides an additional layer of security, as the watermark cannot be read or modified without the appropriate key.

التضمين في المناطق غير المشفرة: يمكن تضمين العلامات المائية في المناطق غير المشفرة من الجينوم (مثل الإنترونات أو المناطق بين الجينات)، حيث لا تؤثر التغييرات على البروتين. هذا النهج يوفر مساحة أكبر للتضمين، لكنه قد لا يكون متاحاً في جميع التسلسلات.

Inheritable Watermarks: Techniques like CentralMark design watermarks that transfer from DNA to the translated protein, allowing detection even after translation. This makes the watermark traceable across the central dogma of molecular biology.

التشفير والتوقيع الرقمي: يمكن تشفير العلامة المائية باستخدام مفاتيح خاصة، وإضافة توقيع رقمي يسمح بالتحقق من أن التسلسل لم يُعدّل. هذا يوفر طبقة إضافية من الأمان، حيث لا يمكن قراءة العلامة المائية أو تعديلها دون المفتاح المناسب.

4. Limits of Watermarks: Not an Independent Security Guarantee

العلامات المائية القابلة للتوريث: تقنيات مثل CentralMark تصمم علامات مائية تنتقل من DNA إلى البروتين المترجم، مما يسمح باكتشافها حتى بعد الترجمة. هذا يجعل العلامة المائية قابلة للتتبع عبر العقيدة المركزية للبيولوجيا الجزيئية.

A watermark alone is not an independent security guarantee. There are several challenges and limitations that must be clear:

4. حدود العلامات المائية: ليست ضماناً أمنياً مستقلاً

Removability or Modifiability: Watermarks can be removed, modified, or pasted into different sequences. If an attacker has sufficient knowledge, they can locate the watermark and remove it, or transfer it to another sequence.

العلامة المائية وحدها ليست ضماناً أمنياً مستقلاً. هناك عدة تحديات وقيود يجب أن تكون واضحة:

Natural Mutations and Synthesis Errors: DNA is susceptible to natural mutations, synthesis errors, and sequencing inaccuracies, which can obscure or degrade watermarks. Even with error correction mechanisms, the watermark may be lost over time or under certain conditions.

قابلية الإزالة أو التعديل: العلامات المائية يمكن إزالتها أو تعديلها أو لصقها في تسلسلات مختلفة. إذا كان لدى المهاجم معرفة كافية، يمكنه تحديد موقع العلامة المائية وإزالتها، أو نقلها إلى تسلسل آخر.

Context Dependence: The watermark alone does not tell you exactly who created the sequence, when, or for what purpose, without additional context (such as laboratory records, or identity management systems). Recognizing a pattern is not proof of intent or responsibility.

الطفرات الطبيعية وأخطاء التوليف: DNA عرضة للطفرات الطبيعية، وأخطاء التوليف، وأخطاء التسلسل، والتي يمكن أن تحجب أو تتلف العلامات المائية. حتى مع آليات تصحيح الأخطاء، قد تفقد العلامة المائية بمرور الوقت أو في ظروف معينة.

Balancing Security and Function: Embedding a watermark may affect the sequence's function if not carefully implemented. Changes may alter protein structure, affect expression, or cause unintended effects. Therefore, watermarks must be carefully designed to avoid affecting biological function.

الاعتماد على السياق: العلامة المائية وحدها لا تخبرك بمن أنشأ التسلسل بالضبط، أو متى، أو لأي غرض، دون سياق إضافي (مثل سجلات المختبر، أو أنظمة إدارة الهوية). التعرف على نمط ليس دليلاً على النية أو المسؤولية.

5. True Traceability: Beyond the Watermark

الموازنة بين الأمان والوظيفة: تضمين علامة مائية قد يؤثر على وظيفة التسلسل إذا لم يُنفذ بعناية. التغييرات قد تغير بنية البروتين، أو تؤثر على التعبير، أو تسبب تأثيرات غير متوقعة. لذلك، يجب تصميم العلامات المائية بعناية لتجنب التأثير على الوظيفة البيولوجية.

True traceability does not rely on the watermark alone. It requires an integrated system including:

5. التتبع الحقيقي: ما وراء العلامة المائية

Identity Management: Each sequence must be associated with a unique and documented identity, not just a recognizable pattern. Identity management includes recording who created the sequence, when, and under what conditions, with mechanisms to verify this information.

التتبع الحقيقي لا يعتمد على العلامة المائية وحدها. يحتاج إلى نظام متكامل يشمل:

Key Management: If encryption is used to protect the watermark, keys must be managed securely. Losing the key means losing the ability to read or verify the watermark. Keys must be protected, stored securely, and available only to authorized individuals.

إدارة الهوية (Identity Management) : يجب أن يكون كل تسلسل مرتبطاً بهوية فريدة وموثقة، وليس فقط بنمط قابل للتعرف. إدارة الهوية تشمل تسجيل من أنشأ التسلسل، ومتى، وتحت أي ظروف، مع آليات للتحقق من هذه المعلومات.

Provenance Records: There must be reliable and tamper-proof records tracking the sequence's history: who created it, who synthesized it, who used it, and when. These records complement the watermark and provide the necessary context for attribution.

إدارة المفاتيح (Key Management) : إذا استُخدم التشفير لحماية العلامة المائية، يجب إدارة المفاتيح بشكل آمن. فقدان المفتاح يعني فقدان القدرة على قراءة العلامة المائية أو التحقق منها. يجب أن تكون المفاتيح محمية، ومخزنة بشكل آمن، ومتاحة فقط للأشخاص المخولين.

Legal and Technical Context: Effective traceability requires a legal framework that supports attribution, and technical mechanisms to verify the authenticity of sequences and their records. Without this framework, the watermark may be just a technical curiosity without practical benefit.

سجلات المصدر (Provenance Records) : يجب أن تكون هناك سجلات موثوقة وغير قابلة للتلاعب تتبع تاريخ التسلسل: من أنشأه، ومن توليفه، ومن استخدمه، ومتى. هذه السجلات تكمل العلامة المائية، وتوفر السياق اللازم لإسناد المسؤولية.

السياق القانوني والتقني: التتبع الفعال يتطلب إطاراً قانونياً يدعم إسناد المسؤولية، وآليات تقنية للتحقق من صحة التسلسلات وسجلاتها. بدون هذا الإطار، قد تكون العلامة المائية مجرد فضول تقني دون فائدة عملية.

التتبع الحقيقي = علامة مائية + إدارة هوية + إدارة مفاتيح + سجلات مصدر + سياق قانوني

6. Practical Application: Educational Simulation of Watermarking

In the educational context, watermark embedding can be simulated using synthetic data, without actual implementation. The goal is to teach principles: how to embed information in a sequence without changing function, and how to extract it.

6. التطبيق العملي: محاكاة تعليمية للعلامة المائية

في السياق التعليمي، يمكن محاكاة تضمين علامة مائية باستخدام بيانات اصطناعية، دون تنفيذ فعلي. الهدف هو تعليم المبادئ: كيفية تضمين معلومات في تسلسل دون تغيير الوظيفة، وكيفية استخراجها.

```
# محاكاة تعليمية لتضمين علامة مائية في تسلسل DNA
def embed_watermark(sequence, watermark, codon_table):
    """
```

محاكاة بسيطة لتضمين علامة مائية باستخدام استبدال الكودونات.

للاستخدام التعليمي فقط (بيانات اصطناعية).

```
    """
    # تبسيط: استبدال الكودون الثالث في كل كودون (محاكاة)
    # في الواقع، يتم اختيار كودونات مترادفة محددة
    result = []
    for i in range(0, len(sequence), 3):
        codon = sequence[i:i+3]
        if len(codon) == 3:
            # محاكاة: تغيير القاعدة الثالثة بناءً على العلامة المائية
            # هذا مجرد مثال تعليمي مبسط
            result.append(codon)
        else:
            result.append(codon)
    return "".join(result)
def extract_watermark(sequence):
    """
```

محاكاة بسيطة لاستخراج علامة مائية.

للاستخدام التعليمي فقط.

```
    """
    في الواقع، يتم فك تشفير العلامة المائية باستخدام مفتاح #
    return "محاكاة: تم استخراج العلامة المائية"

# مثال على الاستخدام
sequence = "ATGGCTAGCTAG"
watermark = "001"
print(f"التسلسل الأصلي: {sequence}")
watermarked = embed_watermark(sequence, watermark, {})
print(f"التسلسل بعد العلامة المائية: {watermarked}")
print(f"العلامة المائية المستخرجة: {extract_watermark(watermarked)}")
```

الدوال المعرفة هنا (embed_watermark, extract_watermark) تمثل حدودًا واضحة بين

المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات

التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational simulation illustrates the principle: information can be embedded in a DNA sequence without changing the apparent function, and extracted later. However, it does not replace a real system with encryption and key management.

7. Watermarks in the Biosecurity Context: Lessons from NIST

هذه المحاكاة التعليمية توضح المبدأ: يمكن تضمين معلومات في تسلسل DNA دون تغيير الوظيفة الظاهرية،

واستخراجها لاحقاً. لكنها لا تغني عن نظام حقيقي مع تشفير وإدارة مفاتيح.

The National Institute of Standards and Technology (NIST) recognizes the importance of watermarks and tracking tools in biosecurity. NIST is working on developing standards for synthetic nucleic acid sequence screening, and providing tools to identify, track, and defend against emerging sequences of concern. NIST also contributes to ISO standards for biosafety and biosecurity.

7. العلامات المائية في سياق الأمن الحيوي: دروس من NIST

One lesson from NIST's work is that watermarks alone are not sufficient; they must be part of a broader system including:

يدرك المعهد الوطني للمعايير والتقنية (NIST) أهمية العلامات المائية وأدوات التتبع في الأمن الحيوي. يعمل NIST

على تطوير معايير لفحص تسلسلات الأحماض النووية الاصطناعية، وتوفير أدوات لتحديد وتتبع وتسلسلات "محل

الاهتمام" الناشئة. كما يساهم NIST في معايير ISO للسلامة الحيوية والأمن الحيوي.

أحد الدروس المستفادة من عمل NIST هو أن العلامات المائية وحدها ليست كافية؛ يجب أن تكون جزءاً من نظام

أوسع يشمل:

· قواعد بيانات مرجعية محدثة: للتعرف على التسلسلات المعروفة والجديدة.

· Updated reference databases: to recognize known and new sequences.

· أدوات فحص قائمة على الوظيفة: لاكتشاف التسلسلات التي قد تكون خطيرة حتى لو لم تشبه أي شيء معروف.

· Function-based screening tools: to detect sequences that may be dangerous even if they resemble nothing known.

· معايير قابلة للتشغيل البيني: لضمان أن الأنظمة المختلفة يمكنها تبادل المعلومات والتعاون.

· Interoperable standards: to ensure different systems can exchange information and collaborate.

· المراجعة البشرية: للتعامل مع الحالات غير المتوقعة والملتبسة.

· Human review: to handle unexpected and ambiguous cases.

Chapter Summary

خلاصة الفصل

DNA watermarks are a mechanism for embedding identifying information in DNA sequences without affecting their biological function. They are used for authorship attribution, traceability, intellectual property protection, and biosecurity. But a watermark alone is not an independent security guarantee; it is removable or modifiable, and may be damaged by mutations or synthesis errors. True traceability requires an integrated system including identity management, key management, provenance records, and legal context. The watermark is a useful tool, but it is not a substitute for comprehensive governance and human review. The next chapter will move to Biosafety Codon Filtering as an additional defensive mechanism.

العلامات المائية للـ DNA هي آلية لتضمين معلومات تعريفية في تسلسلات DNA دون التأثير على وظيفتها البيولوجية. تُستخدم لإثبات التأليف، والتتبع، وحماية الملكية الفكرية، والأمن الحيوي. لكن العلامة المائية وحدها ليست ضماناً أمنياً مستقلاً؛ فهي قابلة للإزالة أو التعديل، وقد تتلف بسبب الطفرات أو أخطاء التوليف. التتبع الحقيقي يتطلب نظاماً متكاملاً يشمل إدارة الهوية، وإدارة المفاتيح، وسجلات المصدر، والسياق القانوني. العلامة المائية هي أداة مفيدة، لكنها ليست بديلاً عن الحوكمة الشاملة والمراجعة البشرية. الفصل القادم سينتقل إلى Biosafety Codon Filtering كآلية دفاعية إضافية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "DNA Watermarking" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "DNA Watermarking" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "DNA watermarks guarantee that sequences will not be misused." This is incorrect; watermarks are a traceability tool, but they are not an independent security guarantee, and can be removed or modified. The correct statement is: "DNA watermarks are a mechanism for traceability and provenance, but they require an integrated system of identity management, key management, and provenance records to be effective, and cannot alone prevent misuse." A simple test: Practical check: "Does this watermark prove who created the sequence, or is it just a pattern that can be imitated?".

مثال على التجاوز: القول إن "العلامات المائية للـ DNA تضمن عدم إساءة استخدام التسلسلات". هذا غير صحيح؛ العلامات المائية هي أداة للتتبع، لكنها ليست ضماناً أمنياً مستقلاً، ويمكن إزالتها أو تعديلها. الصحيح هو: "العلامات المائية للـ DNA هي آلية للتتبع وإثبات المنشأ، لكنها تحتاج إلى نظام متكامل من إدارة الهوية والمفاتيح وسجلات المصدر لتكون فعالة، ولا يمكنها وحدها منع إساءة الاستخدام." اختبار بسيط: فحص تطبيقي: "هل هذه العلامة المائية تثبت من أنشأ التسلسل، أم أنها مجرد نمط يمكن تقليده؟"ـ

Example: The presence of a watermark in a DNA sequence is a computational representation (an embedded pattern). Claiming that "this sequence was created by laboratory X" is a biological/legal claim not proven by the watermark alone. The watermark tells us a pattern exists, but proving that this pattern represents a real identity and responsibility requires additional context (such as documented records, and key management). Practical test: before attributing a sequence to a specific source based on a watermark, verify the context and additional evidence.

مثال: وجود علامة مائية في تسلسل DNA هو تمثيل حسابي (نمط مضمن). الادعاء بأن "هذا التسلسل صُنع بواسطة المختبر X" هو ادعاء بيولوجي/قانوني لا يثبته وجود العلامة المائية وحدها. العلامة المائية تخبرنا بوجود نمط، لكن إثبات أن هذا النمط يمثل هوية حقيقية ومسؤولية يحتاج إلى سياق إضافي (مثل سجلات موثقة، وإدارة مفاتيح). اختبار عملي: قبل أن تعزو تسلسلاً إلى مصدر معين بناءً على علامة مائية، تحقق من السياق والأدلة الإضافيةـ

مثال تطبيقي على نظام العلامات المائية:

· المدخلات: تسلسل DNA أصلي، معلومات تعريفية (مثل معرف المؤلف، التاريخ)، خوارزمية تضمين، مفتاح تشفير (اختياري)ـ

· الافتراضات: أن التغييرات المضمنة لا تؤثر على وظيفة التسلسل، وأن المفتاح محفوظ بشكل آمن، وأن خوارزمية التضمين مقاومة للطفرات والأخطاءـ

· المخرجات: تسلسل DNA معدّل يحوي العلامة المائية.

· القيود: العلامة المائية قد تتلف بسبب الطفرات أو أخطاء التوليف، وقد تُزال أو تُعدّل من قبل جهة خبيثة، ولا توفر سياقاً كاملاً للمسؤولية بدون سجلات إضافية.

Applied example of a watermarking system:

اختبار: قبل الاعتماد على علامة مائية، تحقق من قوة خوارزمية التضمين، وآليات تصحيح الأخطاء، وسلامة إدارة المفاتيح.

· Inputs: original DNA sequence, identifying information (e.g., author ID, date), embedding algorithm, encryption key (optional).

· Assumptions: that the embedded changes do not affect sequence function, that the key is securely stored, and that the embedding algorithm is resistant to mutations and errors.

· Outputs: a modified DNA sequence containing the watermark.

· Limitations: the watermark may be damaged by mutations or synthesis errors, may be removed or modified by a malicious party, and does not provide complete context for accountability without additional records.

Test: before relying on a watermark, verify the strength of the embedding algorithm, error correction mechanisms, and the integrity of key management.

عند استخدام العلامات المائية، يجب تسجيل ما يلي:

· المصدر: التسلسل الأصلي ومصدره، والمعلومات المضمنة في العلامة المائية.

· الإصدار: إصدار خوارزمية التضمين، وإصدار أي مكتبات أو أدوات مستخدمة.

· المعلمات: معاملات التضمين (مثل مواقع التغييرات، قوة التضمين)، والمفتاح المستخدم (إذا كان مشفراً).

· التحويلات: أي خطوات لتنقية البيانات أو تحويلها قبل أو بعد التضمين.

When using watermarks, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لمراجع بعد عام، هل يستطيع فهم كيف ضُمنت العلامة المائية، واستخراجها، والتحقق من صحتها؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the original sequence and its source, and the information embedded in the watermark.

· Version: the version of the embedding algorithm, and any libraries or tools used.

· Parameters: embedding parameters (e.g., change locations, embedding strength), and the key used (if encrypted).

· Transformations: any steps to clean or transform data before or after embedding.

Test: Practical check: "If I gave this record to a reviewer a year later, could they understand how the watermark was embedded, extract it, and verify its authenticity?" If the answer is no, the documentation is insufficient.

قبل تطبيق أي نظام علامات مائية على بيانات حقيقية، اختبره على تسلسلات اصطناعية ذات معلومات معروفة. أنشئ تسلسلاً صغيراً، وأضف علامة مائية تحتوي على معلومات وهمية. حاول استخراج العلامة المائية والتحقق منها. ثم اختبر متانة النظام: أضف طفرات اصطناعية (محاكاة لأخطاء التوليف)، وتأكد من أن العلامة المائية لا تزال قابلة للاستخراج (بفضل آليات تصحيح الأخطاء). إذا نجح الاختبار، فأنت واثق من أن النظام يعمل كما هو متوقع، وأي حالة مع بيانات حقيقية تعكس تعقيدات العالم الحقيقي التي قد تتطلب تحسينات إضافية.

Before applying any watermarking system to real data, test it on synthetic sequences with known information. Create a small sequence, and add a watermark containing dummy information. Try to extract and verify the watermark. Then test the system's robustness: add synthetic mutations (simulating synthesis errors), and verify that the watermark remains extractable (thanks to error correction mechanisms). If the test succeeds, the workflow is confident that the system works as expected, and any case with real data reflects real-world complexities that may require additional improvements.

Chapter 36 — Biosafety Codon Filtering

الفصل

الفكرة المركزية: فكرة الفلترة الدفاعية، قواعد معلنة، false positives، عدم استخدامها للتحايل.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Safety-oriented codon filtering shows how a safety rule can become a testable condition. It is not sufficient by itself because risk can depend on context, full sequence, source, function, and environment. Filters are therefore treated as one defensive layer within a broader system.

الفلترة الوقائية على مستوى الكودونات توضح كيف يمكن تحويل قاعدة سلامة إلى شرط قابل للاختبار. لكنها لا تكفي وحدها؛ لأن الخطر قد يعتمد على السياق، والتسلسل الكامل، والمصدر، والوظيفة، والبيئة. لذلك تُعامل الفلاتر كطبقة دفاعية ضمن نظام أوسع.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. What Is Defensive Codon Filtering؟

1. ما هي فلترة الكودونات الدفاعية؟

Defensive codon filtering is a software mechanism aimed at preventing the execution or generation of DNA sequences that may pose a biological risk, without relying solely on similarity to known sequences. This mechanism is based on pre-defined rules (policies) that specify what is allowed and what is prohibited. Instead of trying to optimize or modify the sequence, the system rejects it and escalates it to human review if it violates any of the declared rules.

فلترة الكودونات الدفاعية هي آلية برمجية تهدف إلى منع تنفيذ أو توليد تسلسلات DNA قد تشكل خطراً بيولوجياً، دون الحاجة إلى الاعتماد فقط على التشابه مع التسلسلات المعروفة. تقوم هذه الآلية على قواعد محددة مسبقاً (سياسات) تحدد ما هو مسموح به وما هو ممنوع. بدلاً من محاولة تحسين التسلسل أو تعديله، يقوم النظام برفضه وإحالته إلى المراجعة البشرية إذا انتهك أي من القواعد المعلنة.

مثال:

قاعدة: لا يُسمح بتسلسلات تحتوي على كودونات معينة مرتبطة بتوليد سموم.

النظام يرفض أي تسلسل يحتوي على هذه الكودونات، حتى لو كان التسلسل لا يشبه أي سم معروف في قاعدة البيانات.

This approach differs from traditional similarity-based screening, which looks for matches with known sequences. Defensive filtering relies on prior knowledge of patterns or properties that pose a risk, regardless of overall similarity.

2. The Algorithm as a Refusal Mechanism, Not Optimization

هذا النهج يختلف عن الفحص التقليدي القائم على التشابه، حيث يبحث عن تطابق مع تسلسلات معروفة. الفلترة الدفاعية تعتمد على معرفة مسبقة بالأنماط أو الخصائص التي تشكل خطراً، بغض النظر عن التشابه الكلي.

In synthetic biology, algorithms are often designed to optimize performance: increase protein efficiency, improve stability, reduce unwanted interactions. But in a security context, these algorithms can be redirected towards refusal. Instead of asking "how do we make this sequence work better?", we ask a different question: "does this sequence violate safety policies?". If the answer is yes, it is rejected or escalated for review.

2. الخوارزمية كآلية رفض، وليس تحسين

في البيولوجيا التركيبية، تُصمم الخوارزميات غالباً لتحسين الأداء: زيادة كفاءة البروتين، تحسين الاستقرار، تقليل التفاعلات غير المرغوب فيها. لكن في السياق الأمني، يمكن إعادة توجيه هذه الخوارزميات نحو الرفض. بدلاً من سؤال "كيف نجعل هذا التسلسل يعمل بشكل أفضل؟"، نطرح سؤالاً مختلفاً: "هل هذا التسلسل يخالف سياسات السلامة؟". إذا كانت الإجابة نعم، يُرفض أو يُصعد للمراجعة.

```
# مثال تعليمي لقاعدة رفض بسيطة
def codon_filter(sequence, forbidden_codons, policy_name="default"):
    """
```

فلترة دفاعية بسيطة: ترفض التسلسل إذا احتوى على كودونات ممنوعة.

```
    """
    for i in range(0, len(sequence), 3):
        codon = sequence[i:i+3]
        if codon in forbidden_codons:
            return {
                "action": "reject",
                "reason": f"تحتوي على كودون ممنوع: {codon}",
                "policy": policy_name,
                "timestamp": "2025-01-15T12:00:00"
            }
    return {
        "action": "approve",
        "reason": "جميع الكودونات مسموحة",
        "policy": policy_name,
        "timestamp": "2025-01-15T12:00:00"
    }
```

الدوال المعرفة هنا (codon_filter) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational example shows that the algorithm does not try to change the sequence, but evaluates it according to declared rules, and decides to reject or approve.

3. The Trade-off Between Sensitivity and Specificity: False Positives and False Negatives

هذا المثال التعليمي يوضح أن الخوارزمية لا تحاول تغيير التسلسل، بل تقيمه وفقاً لقواعد معلنة، وتقرر الرفض أو الموافقة.

In any filtering system, there is a trade-off between sensitivity (the ability to detect dangerous sequences) and specificity (the ability not to classify safe sequences as dangerous). In a defensive context, the most dangerous type of error is a false negative (missing a dangerous sequence). Therefore, defensive systems tend to increase sensitivity, even if this leads to false positives (rejecting safe sequences). False positives can be addressed by human review, while false negatives may lead to irreparable consequences.

3. المفاضلة بين الحساسية والنوعية: الإيجابيات والسلبيات الكاذبة

في أي نظام فلترة، هناك مفاضلة بين الحساسية (القدرة على اكتشاف التسلسلات الخطيرة) والنوعية (القدرة على عدم تصنيف التسلسلات الآمنة على أنها خطيرة). في السياق الدفاعي، نوع الخطأ الأكثر خطورة هو السلبية الكاذبة (تفويت تسلسل خطير). لذلك، تميل الأنظمة الدفاعية إلى زيادة الحساسية، حتى لو أدى ذلك إلى إيجابيات كاذبة (رفض تسلسلات آمنة). الإيجابيات الكاذبة يمكن معالجتها بالمراجعة البشرية، أما السلبيات الكاذبة فقد تؤدي إلى عواقب لا يمكن إصلاحها.

مقارنة:

نظام حساس جداً: يلتقط 99% من الخطورة، لكنه يرفض 30% من الآمنة (كثير من الإيجابيات الكاذبة).

نظام نوعي جداً: يرفض 2% فقط من الآمنة، لكنه يلتقط 60% فقط من الخطورة (كثير من السلبيات الكاذبة).

النظام الدفاعي الأميل للسلامة: يفضل الحساسية العالية، ويعتمد على المراجعة البشرية للإيجابيات الكاذبة.

4. Updating Policy Without Turning It into a Recipe

Security policies are not static. With new sequences, new scientific evidence, and evolving threats, policies must be updated regularly. However, policy updates must be thoughtful, so they do not turn into a "recipe" that can be exploited to bypass the system. Declared policies must be clear enough to implement, but remain flexible enough to handle unexpected cases. In this context, filtering systems are designed to facilitate policy updates, but they do not provide malicious actors with a clear map of how to evade filtering.

4. تحديث السياسة دون تحويلها إلى وصفة

السياسات الأمنية ليست ثابتة. مع ظهور تسلسلات جديدة، وأدلة علمية جديدة، وتطور التهديدات، يجب تحديث السياسات بانتظام. لكن تحديث السياسة يجب أن يكون مدروساً، حتى لا يتحول إلى "وصفة" يمكن استغلالها لتجاوز النظام. السياسات المعلنة يجب أن تكون واضحة بما يكفي لتطبيقها، لكنها يجب أن تبقى مرنة بما يسمح بالتعامل مع الحالات غير المتوقعة. في هذا السياق، يتم تصميم أنظمة الفلترة بحيث تسهل تحديث السياسات، لكنها لا تقدم للجهات الخبيثة خريطة واضحة لكيفية التهرب من الفلترة.

مبدأ:

- السياسات تُعلن للشفافية ولتوجيه المراجعة البشرية.

- لا تُعلن تفاصيل التنفيذ الدقيقة (مثل الخوارزميات الدقيقة، أو قائمة الكودونات الممنوعة بالكامل).

- التحديثات تتم بشكل منتظم، وتوثق في سجلات التدقيق.

5. The Risk of Over-Rejection

5. خطر الرفض الزائد (Over-Rejection)

الرفض الزائد هو عندما يرفض النظام تسلسلات آمنة بكميات كبيرة، مما يعيق البحث العلمي المشروع ويؤدي إلى إهدار الوقت والموارد. في الأنظمة الدفاعية، يجب موازنة الحساسية مع الحد من الرفض الزائد. يمكن تحقيق ذلك من خلال:

Over-rejection occurs when the system rejects safe sequences in large numbers, hindering legitimate scientific research and wasting time and resources. In defensive systems, sensitivity must be balanced with limiting over-rejection. This can be achieved through:

· تحسين القواعد: جعل القواعد أكثر دقة، بحيث تستهدف الخصائص الخطيرة بدلاً من الأنماط العامة.

· Rule refinement: making rules more precise, targeting dangerous properties rather than general patterns.

· المراجعة البشرية الفعالة: وجود آلية سريعة وفعالة للمراجعة البشرية للتعامل مع الإيجابيات الكاذبة.

· Effective human review: having a fast and effective mechanism for human review to handle false positives.

· التغذية الراجعة: استخدام نتائج المراجعة البشرية لتحسين القواعد وتقليل الإيجابيات الكاذبة بمرور الوقت.

· Feedback: using the results of human review to improve rules and reduce false positives over time.

6. Practical Application: Building a Small Defensive Filter

Let's apply these principles in a simple educational example: a codon filtering system that rejects any sequence containing prohibited codons, and records every decision in an audit log.

6. التطبيق العملي: بناء فلترة دفاعية صغيرة

دعنا نطبق هذه المبادئ في مثال تعليمي بسيط: نظام فلترة كودونات يرفض أي تسلسل يحتوي على كودونات ممنوعة، ويسجل كل قرار في سجل تدقيق.

```
#نظام فلترة كودونات دفاعي تعليمي
class CodonFilter:
    def __init__(self, forbidden_codons, policy_name="default"):
        self.forbidden_codons = set(forbidden_codons)
        self.policy_name = policy_name
        self.audit_log = []
def filter_sequence(self, sequence, user_id):
        """
```

تطبيق الفلترة على تسلسل، مع تسجيل القرار في سجل التدقيق.

```
        """
        decision = {
                "user": user_id,
            "sequence": sequence[:20] + "..." if len(sequence) > 20 else sequence,
                "timestamp": "2025-01-15T12:00:00",
                "policy": self.policy_name
                }
#فحص الكودونات الممنوعة
        for i in range(0, len(sequence), 3):
            codon = sequence[i:i+3]
            if codon in self.forbidden_codons:
                decision["action"] = "reject"
                decision["reason"] = f"يحتوي على كودون ممنوع: {codon}"
                self.audit_log.append(decision)
                return decision
        decision["action"] = "approve"
        decision["reason"] = "جميع الكودونات مسموحة"
        self.audit_log.append(decision)
        return decision
    def get_audit_log(self):
        """إرجاع سجل التدقيق."""
        return self.audit_log
```

```
تسلسلات اختبار (اصطناعية) #
sequences = [
    {"seq": "ATGGCGTAA", "user": "researcher_01"},
    {"seq": "ATGGCTAGC", "user": "researcher_02"},
    {"seq": "ATGGCCAAC", "user": "researcher_03"},
]

# مثال على الاستخدام (بيانات اصطناعية)
filter_system = CodonFilter(
    forbidden_codons=["GCT", "GCC", "GCA", "GCG"],  # كودونات ممنوعة (مثال فقط)
    policy_name="biosafety_policy_v1"
)

for s in sequences:
    result = filter_system.filter_sequence(s["seq"], s["user"])
    print(f"{s['user']}: {result['action']} - {result['reason']}")
print("\nسجل التدقيق:")
for entry in filter_system.get_audit_log():
    print(f"  {entry['user']}: {entry['action']} ({entry['reason']})")
```

الدوال المعرفة هنا (__init__, filter_sequence, get_audit_log) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (CodonFilter) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة CodonFilter يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

This educational example demonstrates simple defensive codon filtering with an audit log. However, it does not cover all security complexities, nor does it constitute operational guidance for biological optimization.

7. Filtering in a Broader Context

هذا المثال التعليمي يوضح فلترة كودونات دفاعية بسيطة مع سجل تدقيق. لكنه لا يغطي جميع التعقيدات الأمنية، ولا يشكل إرشاداً تشغيلياً للتحسين البيولوجي.

Codon filtering is not a standalone solution, but one layer in a multi-layer security system. It should be integrated with:

7. الفلترة في سياق أوسع

فلترة الكودونات ليست حلاً مستقلاً، بل هي طبقة واحدة في نظام أمني متعدد الطبقات. يجب أن تُدمج مع:

· فحص التسلسل القائم على التشابه: لاكتشاف التسلسلات المعروفة.

· الفحص القائم على الوظيفة: لاكتشاف التسلسلات الجديدة التي قد تكون خطيرة.

· Similarity-based sequence screening: to detect known sequences.

· Function-based screening: to detect new sequences that may be dangerous.

· إدارة السياسات والإصدارات: لتحديث القواعد بانتظام وتوثيقها.

· سجلات التدقيق: لتتبع القرارات وضمان المساءلة.

· Policy and version management: to update and document rules regularly.

· Audit logs: to track decisions and ensure accountability.

Chapter Summary

خلاصة الفصل

Defensive codon filtering is a software mechanism to reject sequences that violate declared safety policies. Instead of optimizing performance, these algorithms aim to refuse or escalate. The system is designed to be conservative: the absence of authorization or trusted reference leads to halting or human review. The trade-off between sensitivity and specificity requires a balance, with sensitivity preferred in defensive contexts, relying on human review for false positives. Policy updates must be thoughtful, without turning them into a recipe that can be exploited. Filtering is one layer in a multi-layer security system, and should be integrated with other tools and audit logs. The next chapter will move to the biobrick_security project as an integrated educational package.

فلترة الكودونات الدفاعية هي آلية برمجية لرفض التسلسلات التي تخالف سياسات السلامة المعلنة. بدلاً من تحسين الأداء، تهدف هذه الخوارزميات إلى الرفع أو التصعيد. النظام مصمم ليكون محافظاً: غياب الإذن أو المرجع الموثوق يؤدي إلى التوقف أو المراجعة البشرية. المفاضلة بين الحساسية والنوعية تتطلب توازناً، مع تفضيل الحساسية في السياقات الدفاعية، مع الاعتماد على المراجعة البشرية للتعامل مع الإيجابيات الكاذبة. تحديث السياسات يجب أن يكون مدروساً، دون تحويلها إلى وصفة يمكن استغلالها. الفلترة هي طبقة واحدة في نظام أمني متعدد الطبقات، ويجب

دمجها مع أدوات أخرى وسجلات تدقيق. الفصل القادم سينتقل إلى مشروع biobrick_security كحزمة تعليمية متكاملة.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Biosafety Codon Filtering" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Biosafety Codon Filtering" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "codon filtering can prevent all dangerous sequences." This is incorrect; filtering is a defensive tool based on declared rules, but it may miss new unknown sequences, and may produce false positives. The correct statement is: "codon filtering is a defensive mechanism that rejects sequences violating declared policies, but it is not a comprehensive solution, and needs to be integrated with other tools and human review." A simple test: Practical check: "Do these rules cover all potential dangerous sequences?".

مثال على التجاوز: القول إن "فلترة الكودونات يمكنها منع جميع التسلسلات الخطيرة". هذا غير صحيح؛ الفلترة هي أداة دفاعية تعتمد على قواعد معلنة، لكنها قد تفوت تسلسلات جديدة غير معروفة، وقد تنتج إيجابيات كاذبة. الصحيح هو: "فلترة الكودونات هي آلية دفاعية ترفض التسلسلات التي تخالف سياسات معلنة، لكنها ليست حلاً شاملاً، وتحتاج إلى دمجها مع أدوات أخرى ومراجعة بشرية." اختبار بسيط: فحص تطبيقي: "هل هذه القواعد تغطي جميع التسلسلات الخطيرة المحتملة؟"۔

Example: A filtering system rejecting a sequence based on prohibited codons is a computational representation of a policy. Claiming that "this sequence poses a biological risk" is a biological claim not proven by the automated rejection alone. Automated rejection is based on specific rules, but the actual risk requires biological analysis and additional context. Practical test: before classifying a sequence as dangerous based on automated filtering, verify the biological context and additional evidence.

مثال: نظام فلترة يرفض تسلسلاً بناءً على كودونات ممنوعة هو تمثيل حسابي لسياسة. الادعاء بأن "هذا التسلسل يشكل خطراً بيولوجياً" هو ادعاء بيولوجي لا يثبته الرفض الآلي وحده. الرفض الآلي يعتمد على قواعد محددة، لكن الخطر الفعلي يحتاج إلى تحليل بيولوجي وسياق إضافي. اختبار عملي: قبل أن تصنف تسلسلاً على أنه خطير بناءً على فلترة آلية، تحقق من السياق البيولوجي والأدلة الإضافية۔

مثال تطبيقي على نظام فلترة كودونات:

· المدخلات: تسلسل DNA، قائمة كودونات ممنوعة، سياسة محددة، هوية المستخدم.

· الافتراضات: أن قائمة الكودونات الممنوعة تعكس المخاطر البيولوجية الحالية، وأن السياسة محدثة، وأن سجل التدقيق غير قابل للتلاعب.

· المخرجات: قرار (رفض، موافقة، تعليق للمراجعة)، وسجل تدقيق محدث.

· القيود: النظام قد يرفض تسلسلات آمنة (إيجابيات كاذبة)، وقد يفوت تسلسلات خطيرة جديدة (سلبيات كاذبة)، ولا يستبدل المراجعة البشرية.

Applied example of a codon filtering system:

اختبار: قبل الاعتماد على نظام فلترة، تحقق من تحديث قائمة الكودونات الممنوعة، وسياسة التعامل مع الحالات الجديدة، وآلية المراجعة البشرية.

· Inputs: DNA sequence, list of prohibited codons, specific policy, user identity.

· Assumptions: that the list of prohibited codons reflects current biological risks, that the policy is up to date, and that the audit log is tamper-proof.

· Outputs: decision (reject, approve, hold for review), and updated audit log.

· Limitations: the system may reject safe sequences (false positives), may miss new dangerous sequences (false negatives), and does not replace human review.

Test: before relying on a filtering system, verify the update of the prohibited codon list, the policy for handling new cases, and the human review mechanism.

عند تصميم أو استخدام نظام فلترة، يجب تسجيل ما يلي:

· المصدر: السياسات والقواعد التي يستند إليها النظام (المراجع القانونية والعلمية).

· الإصدار: إصدار النظام البرمجي، وإصدار قائمة الكودونات الممنوعة.

· المعلمات: قائمة الكودونات الممنوعة، أي معاملات أخرى تؤثر على القرار.

When designing or using a filtering system, record the following:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لمراجع بعد عام، هل يستطيع فهم كيف اتخذت القرارات، وما هي السياسات التي طُبقت، وهل كانت متسقة مع المعايير الأمنية؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the policies and rules the system is based on (legal and scientific references).

· Version: the software system version, and the version of the prohibited codon list.

· Parameters: the list of prohibited codons, and any other parameters affecting decisions.

Test: Practical check: "If I gave this record to a reviewer a year later, could they understand how decisions were made, what policies were applied, and whether they were consistent with security standards?" If the answer is no, the documentation is insufficient.

قبل تطبيق نظام فلترة على بيانات حقيقية، اختبره على تسلسلات اصطناعية ذات خصائص معروفة. أنشئ مجموعة من التسلسلات التي تحتوي على كودونات ممنوعة وأخرى آمنة. شغّل النظام وتأكد من أنه يصنفها بشكل صحيح وفقاً للقواعد المحددة (حساسية ونوعية). اختبر أيضاً حالات حدية (مثل تسلسلات تحتوي على كودونات ممنوعة لكنها قد تكون آمنة في سياق معين) لترى كيف يتعامل النظام مع الغموض. إذا نجح الاختبار، فأنت واثق من أن النظام يعمل كما هو متوقع، وأي حالة مع بيانات حقيقية تعكس تعقيدات العالم الحقيقي التي قد تتطلب مراجعة بشرية.

Before applying a filtering system to real data, test it on synthetic sequences with known properties. Create a set of sequences containing prohibited codons and safe sequences. Run the system and verify that it classifies them correctly according to the specified rules (sensitivity and specificity). Also test edge cases (such as sequences containing prohibited codons but which may be safe in a specific context) to see how the system handles ambiguity. If the test succeeds, the workflow is confident that the system works as expected, and any case with real data reflects real-world complexities that may require human review.

الفكرة المركزية: API، طبقات، الاختبارات، سياسات القرار، السجلات.

Chapter 37 — Engineering the Educational Security Package

الفصل 37 — هندسة الحزمة الأمنية التعليمية

Deeper Conceptual Explanation

شرح أعمق للمفهوم

A well-designed educational package makes safe behavior structural: clear inputs, early validation, safe failure, logging, and synthetic-data tests. The goal is to teach how controls are designed, not to turn the example into a ready-to-deploy tool for a sensitive context.

الحزمة التعليمية الجيدة تجعل السلوك الآمن افتراضاً بنيوياً: مدخلات واضحة، تحقق مبكر، فشل آمن، سجلات، واختبارات على بيانات اصطناعية. الهدف هو تعليم طريقة التفكير في الضوابط، لا تحويل المثال إلى أداة تشغيلية جاهزة لسياق حساس.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Why an Educational Package؟

1. لماذا حزمة تعليمية؟

The educational package is not a ready-to-use tool for a production environment. It is an educational model illustrating the engineering principles of software biosecurity. Using an educational package, the workflow can learn how to organize code, separate responsibilities, write tests, and document decisions, without handling sensitive data or operational policies. The educational package is a safe laboratory for experimenting with governance concepts before applying them in real systems.

الحزمة التعليمية ليست أداة جاهزة للاستخدام في بيئة إنتاجية. هي نموذج تعليمي يوضح المبادئ الهندسية للأمن الحيوي البرمجي. باستخدام حزمة تعليمية، يمكن تعلم كيفية تنظيم الكود، وفصل المسؤوليات، وكتابة اختبارات، وتوثيق القرارات، دون التعامل مع بيانات حساسة أو سياسات تشغيلية. الحزمة التعليمية هي مختبر آمن لتجربة مفاهيم الحوكمة قبل تطبيقها في أنظمة حقيقية.

خصائص الحزمة التعليمية:

- بيانات اصطناعية فقط (لا توجد قواعد بيانات تشغيلية).

- سياسات تعليمية مبسطة.

- اختبارات توضح المبادئ.

- سجلات تدقيق توضيحية.

- واجهات واضحة وقابلة للتوسع.

2. Modular Design

Modular design means dividing the system into independent modules, each with a specific responsibility. This facilitates understanding, testing, and updating. In the biobrick_security educational package, the modules can be divided as follows:

2. التصميم المعياري (Modularity)

التصميم المعياري يعني تقسيم النظام إلى وحدات (modules) مستقلة، لكل منها مسؤولية محددة. هذا يسهل الفهم والاختبار والتحديث. في حزمة biobrick_security التعليمية، يمكن تقسيم الوحدات كالتالي:

· screening.py: فحص التسلسل القائم على التشابه. يحتوي على دوال لمقارنة التسلسلات بقواعد بيانات مرجعية (بيانات اصطناعية).

· off_target.py: تحليل المواقع غير المستهدفة. يحتوي على دوال لتحديد المواقع المحتملة للقطع غير المقصود.

· policy.py: إدارة السياسات. يحتوي على تعريفات السياسات، وقواعد التصنيف، وعتبات القرار.

· watermark.py: العلامات المائية للـ DNA. يحتوي على دوال لتضمين واستخراج العلامات المائية (تعليمي).

· models.py: نماذج البيانات. يحتوي على تعريفات الكائنات المستخدمة عبر الحزمة (مثل طلب، قرار، سجل).

· screening.py: similarity-based sequence screening. Contains functions for comparing sequences against reference databases (synthetic data).

· audit.py: سجلات التدقيق. يحتوي على دوال لتسجيل القرارات، وتخزينها، واسترجاعها.

· off_target.py: off-target analysis. Contains functions for identifying potential unintended cut sites.

· tests/: اختبارات الوحدة. يحتوي على اختبارات لكل وحدة، باستخدام بيانات اصطناعية.

· policy.py: policy management. Contains policy definitions, classification rules, and decision thresholds.

· watermark.py: DNA watermarking. Contains functions for embedding and extracting watermarks (educational).

· models.py: data models. Contains definitions of objects used across the package (e.g., request, decision, log).

· audit.py: audit logs. Contains functions for recording decisions, storing them, and retrieving them.

· tests/: unit tests. Contains tests for each module, using synthetic data.

هيكل الحزمة:

```
biobrick_security/
├── __init__.py
├── screening.py
├── off_target.py
├── policy.py
├── watermark.py
├── models.py
```

```
├── audit.py
└── tests/
    ├── test_screening.py
    ├── test_off_target.py
    └── ...
```

This modular structure allows developing and testing each module independently, and updating policies without affecting analysis logic.

3. Separating Policy from Analysis Logic

هذا الهيكل المعياري يسمح بتطوير واختبار كل وحدة على حدة، وتحديث السياسات دون التأثير على منطق التحليل.

The most important principle in designing security systems is separating policy from analysis logic. Policy is the set of rules that define what is allowed and what is prohibited. Analysis logic is the algorithms that implement these policies. If policy is buried inside an analysis function, updating it becomes difficult, and reviewing it becomes nearly impossible. In good design, policy is defined in a separate module (e.g., policy.py), and passed as a parameter to analysis functions. This allows updating policy without changing the core code, and allows independent policy review.

3. فصل السياسة عن منطق التحليل

أهم مبدأ في تصميم الأنظمة الأمنية هو فصل السياسة عن منطق التحليل. السياسة هي مجموعة القواعد التي تحدد ما هو مسموح وما هو ممنوع. منطق التحليل هو الخوارزميات التي تنفذ هذه السياسات. إذا دُفنت السياسة داخل دالة تحليلية، يصبح تحديثها صعباً، وتصبح مراجعتها مستحيلة تقريباً. في التصميم الجيد، تُعرَّف السياسة في وحدة منفصلة (مثل policy.py)، وتُمرر كمعامل إلى دوال التحليل. هذا يسمح بتحديث السياسة دون تغيير الكود الأساسي، ويسمح بمراجعة السياسة بشكل مستقل.

```
# مثال تعليمي: فصل السياسة عن التحليل
# في policy.py
class ScreeningPolicy:
    def __init__(self, similarity_threshold=0.8, forbidden_codons=None):
        self.similarity_threshold = similarity_threshold
        self.forbidden_codons = forbidden_codons or []
# في screening.py
def screen_sequence(sequence, policy, reference_db):
    # منطق التحليل يستخدم السياسة كمعامل
    for ref in reference_db:
        similarity = calculate_similarity(sequence, ref)
        if similarity >= policy.similarity_threshold:
            return {"action": "review", "reason": "high similarity"}
    return {"action": "approve", "reason": "low similarity"}
```

الدوال المعرفة هنا (__init__, screen_sequence) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (ScreeningPolicy) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة ScreeningPolicy يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

This example shows that the analysis function does not contain fixed policy values; it receives them as a parameter. This makes the code testable, reusable, and easily updatable.

4. Writing Tests and Documenting Releases

هذا المثال يوضح أن دالة التحليل لا تحتوي على القيم الثابتة للسياسة؛ بل تستقبلها كمعامل. هذا يجعل الكود قابلاً للاختبار وإعادة الاستخدام وتحديث السياسة بسهولة.

Tests are an essential part of any software package. In the educational package, tests are written to verify that each module works as expected, that policies are applied correctly, and that audit logs are recorded accurately. Tests are written using synthetic data, so that results are known in advance.

4. كتابة الاختبارات وتوثيق الإصدارات

الاختبارات هي جزء أساسي من أي حزمة برمجية. في الحزمة التعليمية، تُكتب اختبارات للتحقق من أن كل وحدة تعمل كما هو متوقع، وأن السياسات تُطبق بشكل صحيح، وأن سجلات التدقيق تُسجل بدقة. الاختبارات تُكتب باستخدام بيانات اصطناعية، بحيث تكون النتائج معروفة مسبقاً.

```
# مثال: اختبار لوحدة screening
def test_screening_with_policy():
    policy = ScreeningPolicy(similarity_threshold=0.8)
    reference_db = {"toxin_A": "GACTAGCTAGCTAGCTAGC"}
    sequence_high = "GACTAGCTAGCTAGCTAGC"
    sequence_low = "ATGCGTACGTAGCTAGC"
result_high = screen_sequence(sequence_high, policy, reference_db)
    assert result_high["action"] == "review"
```

```
result_low = screen_sequence(sequence_low, policy, reference_db)
    assert result_low["action"] == "approve"
```

الدوال المعرفة هنا (test_screening_with_policy) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها.

Documenting releases means recording each package version, with descriptions of changes, fixes, and updates. This allows tracking developments, and knowing which version was used in any analysis.

توثيق الإصدارات يعني تسجيل كل إصدار من الحزمة، مع وصف التغييرات، والإصلاحات، والتحديثات. هذا يسمح بتتبع التطورات، ومعرفة أي إصدار استُخدم في أي تحليل.

مثال لسجل الإصدارات:

الإصدار 1.0.0 (15-01-2025): الإطلاق الأول، وحدات أساسية للفحص والفلترة.

الإصدار 1.1.0 (20-02-2025): إضافة وحدة العلامات المائية، تحديث السياسات.

الإصدار 1.2.0 (10-03-2025): تحسين الاختبارات، إصلاح أخطاء في سجل التدقيق.

5. Dependency Management

The educational package depends on core Python libraries (such as Biopython, and possibly NumPy for numerical operations). These dependencies must be managed clearly, so that anyone can install and run the package without issues. In Python, a requirements.txt or pyproject.toml file is used to document dependencies and their versions.

5. إدارة الاعتماديات (Dependencies)

الحزمة التعليمية تعتمد على مكتبات Python الأساسية (مثل Biopython، وربما NumPy للعمليات الحسابية). يجب أن تُدار هذه الاعتماديات بشكل واضح، بحيث يمكن لأي شخص تثبيت الحزمة وتشغيلها دون مشاكل. في Python، يُستخدم ملف requirements.txt أو pyproject.toml لتوثيق الاعتماديات وإصداراتها.

مثال لملف requirements.txt:

biopython>=1.81,<2.0

```
numpy>=1.24,<2.0
pytest>=7.0,<8.0
```

Documenting dependencies ensures that the software environment is reproducible, and that any library update does not break the package.

6. Security Layers: Fail-Closed and Human Review

توثيق الاعتماديات يضمن أن البيئة البرمجية قابلة لإعادة الإنتاج، وأن أي تحديث لمكتبة لا يكسر الحزمة.

In high-consequence systems, the default behavior should be to halt or escalate to human review when conditions are absent. This principle (fail-closed) is the essence of the security design of the educational package. In every module of the package, there should be verification of authorization, conditions, and audit log before any sensitive step.

6. طبقات الأمان: الفشل مغلق والمراجعة البشرية

في الأنظمة ذات العواقب العالية، يجب أن يكون السلوك الافتراضي هو التوقف أو التصعيد إلى مراجعة بشرية عند فقدان الشرط. هذا المبدأ (fail-closed) هو جوهر التصميم الأمني للحزمة التعليمية. في كل وحدة من وحدات الحزمة، يجب أن يكون هناك تحقق من الصلاحية، والشروط، وسجل التدقيق قبل أي خطوة حساسة.

```
# تطبيق: مثال تعليمي في وحدة fail-closed gene_drive_lock
def gene_drive_lock(request, policy, audit_log):
    # التحقق من الصلاحية
    if not request.get("approved", False):
        audit_log.append("فشل: لا يوجد موافقة صريحة")
        return {"status": "denied", "reason": "no explicit approval"}
# التحقق من الشروط
    if not policy.check_conditions(request):
        audit_log.append("فشل: الشروط غير مستوفاة")
        return {"status": "pending_review", "reason": "conditions not met"}
# تنفيذ الإجراء (محاكاة)
    return {"status": "executed", "reason": "success"}
```

الدوال المعرفة هنا (gene_drive_lock) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This example shows that any sensitive step requires verification of authorization and conditions, with the decision recorded in the audit log. This is the governance model the educational package provides.

7. Limits of the Educational Model

هذا المثال يوضح أن أي خطوة حساسة تتطلب تحققاً من الصلاحية والشروط، مع تسجيل القرار في سجل التدقيق.

هذا هو نموذج الحوكمة الذي تقدمه الحزمة التعليمية.

7. حدود النموذج التعليمي

الحزمة التعليمية هي نموذج، وليست أداة تشغيلية. حدودها واضحة:

· لا تحتوي على قواعد بيانات حقيقية لمسببات الأمراض أو السموم.

The educational package is a model, not an operational tool. Its limits are clear:

· السياسات مبسطة ولا تعكس التعقيدات القانونية أو الأخلاقية للأنظمة الحقيقية.

· It does not contain real databases of pathogens or toxins.

· الاختبارات تغطي سيناريوهات تعليمية، وليست جميع الحالات الواقعية.

· Policies are simplified and do not reflect the legal or ethical complexities of real systems.

· لا تقدم إرشادات تشغيلية للتحسين البيولوجي.

· Tests cover educational scenarios, not all real-world cases.

· It does not provide operational guidance for biological optimization.

Nevertheless, the educational model provides a solid foundation for understanding the principles of security system engineering, and can be used as a platform to explore governance concepts in a safe environment.

مع ذلك، يوفر النموذج التعليمي أساساً متيناً لفهم مبادئ هندسة الأنظمة الأمنية، ويمكن استخدامه كمنصة

لاستكشاف مفاهيم الحوكمة في بيئة آمنة.

Chapter Summary

خلاصة الفصل

The engineering of the biobrick_security educational package demonstrates how to build a multi-layer security system. Modular design allows developing and testing each module independently. Separating policy from analysis logic facilitates policy updates and review. Testing, documentation, and dependency management are the foundation of reproducibility. Applying the fail-closed principle and human review ensures that sensitive decisions are not made automatically without verification. The educational package is a safe model for exploring governance concepts, not an operational tool. The next chapter will move to the details of the pathogen_check module as an educational tool for defensive screening.

هندسة الحزمة الأمنية التعليمية biobrick_security توضح كيفية بناء نظام أمني متعدد الطبقات. التصميم المعياري يسمح بتطوير واختبار كل وحدة على حدة. فصل السياسة عن منطق التحليل يسهل تحديث السياسات ومراجعتها. الاختبارات والتوثيق وإدارة الاعتماديات هي أساس قابلية إعادة الإنتاج. تطبيق مبدأ fail-closed والمراجعة البشرية يضمن أن القرارات الحساسة لا تُتخذ آلياً دون تحقق. الحزمة التعليمية هي نموذج آمن لاستكشاف مفاهيم الحوكمة، وليست أداة تشغيلية. الفصل القادم سينتقل إلى تفاصيل وحدة pathogen_check كأداة تعليمية للفحص الدفاعي.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Engineering the Educational Security Package" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "هندسة الحزمة الأمنية التعليمية" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "this package can be used in real biosecurity systems." This is incorrect; the package is educational and does not contain operational data or policies. The correct statement is: "this package is an educational model illustrating the principles of security system engineering, and can be used to learn concepts, but it is not ready for deployment in real environments." A simple test: Practical check: "Does this package contain real databases and approved security policies?".

مثال على التجاوز: القول إن "هذه الحزمة يمكن استخدامها في أنظمة الأمن الحيوي الحقيقية". هذا غير صحيح؛ الحزمة تعليمية ولا تحتوي على بيانات أو سياسات تشغيلية. الصحيح هو: "هذه الحزمة هي نموذج تعليمي يوضح مبادئ هندسة الأنظمة الأمنية، ويمكن استخدامها لتعلم المفاهيم، لكنها ليست جاهزة للتطبيق في بيئات حقيقية." اختبار بسيط: فحص تطبيقي: "هل هذه الحزمة تحتوي على قواعد بيانات حقيقية وسياسات أمنية معتمدة؟".

Example: An educational package implementing security policies is a computational representation of those policies. Claiming that "these policies guarantee biosafety" is a biological claim not proven by the educational model. The model demonstrates how to apply policies, but it does not prove their effectiveness in a real biological context. Practical test: before inferring policy effectiveness from an educational model, verify the existence of experimental evidence and real context.

مثال: حزمة تعليمية تطبق سياسات أمنية هي تمثيل حسابي لتلك السياسات. الادعاء بأن "هذه السياسات تضمن السلامة البيولوجية" هو ادعاء بيولوجي لا يثبته النموذج التعليمي. النموذج يوضح كيفية تطبيق السياسات، لكنه لا

يثبت فعاليتها في سياق بيولوجي حقيقي. اختبار عملي: قبل أن تستنتج فعالية سياسة من نموذج تعليمي، تحقق من وجود أدلة تجريبية وسياق حقيقي.

مثال تطبيقي على حزمة biobrick_security التعليمية:

· المدخلات: تسلسل استعلام، سياسة محددة، بيانات اصطناعية مرجعية.

· الافتراضات: أن السياسة تعكس قواعد أمنية معقولة، وأن البيانات الاصطناعية تمثل سيناريوهات تعليمية، وأن الاختبارات تغطي الحالات الأساسية.

· المخرجات: قرار (رفض، موافقة، تعليق)، وسجل تدقيق.

· القيود: الحزمة لا تحتوي على بيانات حقيقية، والسياسات مبسطة، والاختبارات ليست شاملة، ولا تقدم إرشادات تشغيلية.

Applied example of the biobrick_security educational package:

اختبار: قبل استخدام الحزمة لأي غرض، تأكد من فهمك لحدودها، وأنك لا تستخدمها في سياق يتطلب بيانات أو سياسات حقيقية.

· Inputs: query sequence, specific policy, synthetic reference data.

· Assumptions: that the policy reflects reasonable security rules, that the synthetic data represents educational scenarios, and that the tests cover basic cases.

· Outputs: decision (reject, approve, hold), and audit log.

· Limitations: the package does not contain real data, policies are simplified, tests are not exhaustive, and it does not provide operational guidance.

Test: before using the package for any purpose, ensure you understand its limitations, and that the workflow is not using it in a context requiring real data or policies.

عند استخدام الحزمة التعليمية، يجب تسجيل:

· المصدر: الحزمة نفسها (إصدارها، تاريخ تحميلها).

· الإصدار: إصدار الحزمة، وإصدارات الاعتماديات.

· المعلمات: السياسات المستخدمة، وأي معاملات تم تغييرها عن القيم الافتراضية.

· التحويلات: أي تعديلات على الكود أو البيانات قبل الاستخدام.

When using the educational package, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة تشغيل نفس الاختبارات والحصول على نفس النتائج؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the package itself (its version, download date).

· Version: the package version, and dependency versions.

· Parameters: the policies used, and any parameters changed from defaults.

· Transformations: any modifications to the code or data before use.

Test: Practical check: "If I gave this record to a colleague a year later, could they rerun the same tests and obtain the same results?" If the answer is no, the documentation is insufficient.

الحزمة التعليمية نفسها هي مثال على استخدام اختبارات اصطناعية. قبل تطبيق أي نظام أمني على بيانات حقيقية، اختبره على بيانات اصطناعية. استخدم الحزمة التعليمية كمنصة لاختبار سياسات مختلفة، وفهم كيفية عمل النظام في سيناريوهات مختلفة. إذا نجحت الاختبارات على البيانات الاصطناعية، فأنت واثق من أن منطق النظام يعمل كما هو متوقع. وعند الانتقال إلى بيانات حقيقية، تذكر أن البيانات الاصطناعية لا تغطي كل التعقيدات الواقعية، وقد تتطلب تعديلات.

The educational package itself is an example of using synthetic tests. Before applying any security system to real data, test it on synthetic data. Use the educational package as a platform to test different policies, and understand how the system works in different scenarios. If the tests succeed on synthetic data, the workflow is confident that the system logic works as expected. When moving to real data, remember that synthetic data does not cover all real-world complexities, and may require adjustments.

Chapter 38 — pathogen_check

الفصل

الفكرة المركزية: بيانات اصطناعية، مطابقة دفاعية، نتائج تحذيرية، عدم تضمين قوائم خطرة.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

A tool name can imply more capability than an algorithm actually has. A defensive checker should state exactly what it searches for, which database or rules it uses, when it stops, and when it requests review. A negative result does not mean absence of risk, and a positive result is not a final biological verdict.

اسم الأداة قد يوحي بقدرة أكبر من قدرة الخوارزمية الفعلية. الفحص الدفاعي يجب أن يوضح بدقة ما الذي يبحث عنه، وما قاعدة البيانات أو القواعد التي يعتمد عليها، ومتى يتوقف، ومتى يطلب مراجعة. النتيجة السلبية لا تعني غياب الخطر، كما أن النتيجة الإيجابية لا تعني حكماً بيولوجياً نهائياً.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Why pathogen_check؟

1. لماذا pathogen_check؟

pathogen_check is an educational module aimed at simulating the process of screening sequences for potential pathogens, using only synthetic data. The goal is not to identify real pathogens, but to teach principles:

pathogen_check هي وحدة تعليمية تهدف إلى محاكاة عملية فحص التسلسلات للكشف عن مسببات الأمراض المحتملة، باستخدام بيانات اصطناعية فقط. الهدف ليس تحديد مسببات الأمراض الحقيقية، بل تعليم المبادئ:

· كيفية تصميم نظام فحص دفاعي.

· كيفية التعامل مع الإيجابيات الكاذبة والسلبيات الكاذبة.

· How to design a defensive screening system.

· أهمية تحديث قواعد البيانات المرجعية.

· How to handle false positives and false negatives.

· دور المراجعة البشرية في النظام.

· The importance of updating reference databases.

· توثيق القرارات وسجلات التدقيق.

· The role of human review in the system.

· Documenting decisions and audit logs.

ما تقدمه pathogen_check:

- بيانات اصطناعية آمنة للتعليم.

- محاكاة لعملية الفحص.

- تقييم أداء النظام (حساسية، نوعية).

- آلية مراجعة بشرية محاكاة.

- سجل تدقيق لكل قرار.

2. Synthetic Data for Defensive Screening

In pathogen_check, neutral synthetic data is used. This data does not represent any real organism, nor does it carry any dangerous information. The goal is to provide a safe environment for learning screening principles, with complete control over data properties (such as similarity level, presence of certain patterns).

2. البيانات الاصطناعية للفحص الدفاعي

في pathogen_check، تُستخدم بيانات اصطناعية محايدة. هذه البيانات لا تمثل أي كائن حي حقيقي، ولا تحمل أي معلومات خطيرة. الهدف هو توفير بيئة آمنة لتعلم مبادئ الفحص، مع إمكانية التحكم الكامل في خصائص البيانات (مثل مستوى التشابه، وجود أنماط معينة).

```
# بيانات اصطناعية تعليمية لـ pathogen_check
SYNTHETIC_DATABASE = {
    "pathogen_sim_01": "GACTAGCTAGCTAGCTAGCNGG",
    "pathogen_sim_02": "GACTAGCTAGCTAGCTATCNGG",
    "pathogen_sim_03": "ATGCGTACGTAGCTAGCTAGCTAGCTAGC",
    "safe_sim_01": "ATGCGTACGTAGCTAGCTAGCTAGCTAGC",
    "safe_sim_02": "GACTAGCTAGCTAGGTAGCNGG",
}
```

This synthetic data allows testing the system in different scenarios: perfect match, partial similarity, no similarity. It also allows testing the effect of updating the reference by adding or removing sequences.

3. Measuring Classification Errors

هذه البيانات الاصطناعية تسمح باختبار النظام في سيناريوهات مختلفة: تطابق تام، تشابه جزئي، لا تشابه. كما تسمح باختبار تأثير تحديث المرجع بإضافة أو إزالة تسلسلات.

One of the most important goals of pathogen_check is to teach how to measure classification errors. In any screening system, there are two types of errors:

3. قياس أخطاء التصنيف

من أهم أهداف pathogen_check هو تعليم كيفية قياس أخطاء التصنيف. في أي نظام فحص، هناك نوعان من الأخطاء:

False Positive: the system classifies a safe sequence as dangerous.

الإيجابية الكاذبة (False Positive): يصنف النظام تسلسلاً آمناً على أنه خطير.

False Negative: the system classifies a dangerous sequence as safe.

السلبيات الكاذبة (False Negative): يصنف النظام تسلسلاً خطيراً على أنه آمن.

In security contexts, false negatives are often more dangerous than false positives. Therefore, defensive systems tend to increase sensitivity (to reduce false negatives) even if this leads to more false positives.

في السياقات الأمنية، السلبيات الكاذبة غالباً ما تكون أكثر خطورة من الإيجابيات الكاذبة. لذلك، تميل الأنظمة الدفاعية

إلى زيادة الحساسية (لتقليل السلبيات الكاذبة) حتى لو أدى ذلك إلى زيادة الإيجابيات الكاذبة.

```
#مثال: حساب أخطاء التصنيف في بيانات اختبار
def calculate_classification_metrics(results, ground_truth):
    """
```

حساب الإيجابيات الكاذبة والسلبيات الكاذبة.

```
    """
    tp = fp = tn = fn = 0
    for seq_id, result in results.items():
        if result == "pathogen" and ground_truth[seq_id] == "pathogen":
            tp += 1
        elif result == "pathogen" and ground_truth[seq_id] == "safe":
            fp += 1
        elif result == "safe" and ground_truth[seq_id] == "safe":
            tn += 1
        elif result == "safe" and ground_truth[seq_id] == "pathogen":
            fn += 1
    return {"tp": tp, "fp": fp, "tn": tn, "fn": fn}
```

الدوال المعرفة هنا (calculate_classification_metrics) تمثل حدودًا واضحة بين المدخلات

والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل

إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational example shows how to measure system performance, which is fundamental for evaluating any screening system.

4. Designing a Human Review Queue

هذا المثال التعليمي يوضح كيفية قياس أداء النظام، وهو أساسي لتقييم أي نظام فحص.

In pathogen_check, the final decision is not made automatically in unclear cases. Instead, these cases are sent to a human review queue. This simulates real systems where high-risk or uncertain cases require human intervention.

4. تصميم قائمة انتظار المراجعة البشرية

في pathogen_check، لا يُتخذ القرار النهائي آلياً في الحالات غير الواضحة. بدلاً من ذلك، تُرسل هذه الحالات إلى قائمة انتظار المراجعة البشرية. هذا يحاكي الأنظمة الحقيقية حيث تتطلب الحالات عالية المخاطر أو غير المؤكدة تدخلاً بشرياً.

```
# مثال: قائمة انتظار مراجعة بشرية محاكاة
class HumanReviewQueue:
    def __init__(self):
        self.queue = []
        self.reviewed = []
def review_next(self, reviewer_id, decision, rationale):
    """مراجعة الحالة التالية في قائمة الانتظار."""
    if not self.queue:
        return None
    case = self.queue.pop(0)
```

```
case["reviewer"] = reviewer_id
case["decision"] = decision
case["rationale"] = rationale
case["review_date"] = "2025-01-15"
```

```
    self.reviewed.append(case)
    return case
def add_for_review(self, case):
    """إضافة حالة للمراجعة."""
    self.queue.append(case)
```

الدوال المعرفة هنا (__init__, add_for_review, review_next) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (HumanReviewQueue) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة HumanReviewQueue يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

This educational model illustrates the human review mechanism, where the reviewer's identity, decision, rationale, and date are recorded. This ensures auditability and accountability.

5. Practical Application: Running pathogen_check

هذا النموذج التعليمي يوضح آلية المراجعة البشرية، حيث تُسجل هوية المراجع، والقرار، والأساس المنطقي، والتاريخ.

هذا يضمن قابلية التدقيق والمساءلة.

Let us apply pathogen_check in a simple educational example. We will screen several synthetic sequences, classify them, and send unclear cases for human review.

5. التطبيق العملي: تشغيل pathogen_check

دعنا نطبق pathogen_check في مثال تعليمي بسيط. سنقوم بفحص عدة تسلسلات اصطناعية، وتصنيفها،

وإرسال الحالات غير الواضحة للمراجعة البشرية.

```
# محاكاة تعليمية لـ pathogen_check
class PathogenCheck:
    def __init__(self, reference_db, similarity_threshold=0.8):
        self.reference_db = reference_db
        self.similarity_threshold = similarity_threshold
        self.review_queue = HumanReviewQueue()
        self.audit_log = []
def check_sequence(self, sequence, seq_id):
        """فحص تسلسل وتصنيفه."""
        # حساب أقصى تشابه مع قاعدة البيانات
        max_similarity = 0
        for ref_id, ref_seq in self.reference_db.items():
            matches = sum(1 for a, b in zip(sequence, ref_seq) if a == b)
            similarity = matches / len(sequence) if len(sequence) > 0 else 0
            max_similarity = max(max_similarity, similarity)
# تصنيف بناءً على التشابه
        if max_similarity >= self.similarity_threshold:
            classification = "potential_pathogen"
            action = "review"
        else:
            classification = "safe"
            action = "approve"
# تسجيل القرار
        decision = {
```

```
"seq_id": seq_id,
"max_similarity": max_similarity,
"classification": classification,
"action": action,
"timestamp": "2025-01-15T12:00:00"
```

```
}
        self.audit_log.append(decision)
        # إرسال للمراجعة إذا لزم الأمر
        if action == "review":
            self.review_queue.add_for_review(decision)
        return decision
    def get_audit_log(self):
        return self.audit_log
    def get_review_queue(self):
        return self.review_queue.queue
# مثال على الاستخدام
db = {
    "pathogen_sim_01": "GACTAGCTAGCTAGCTAGCNGG",
    "pathogen_sim_02": "GACTAGCTAGCTAGCTATCNGG",
}
checker = PathogenCheck(db, similarity_threshold=0.8)
test_sequences = [
    {"id": "seq_001", "seq": "GACTAGCTAGCTAGCTAGCNGG"},  # تطابق تام
    {"id": "seq_002", "seq": "GACTAGCTAGCTAGCTATCNGG"},  # تطابق تام
    {"id": "seq_003", "seq": "ATGCGTACGTAGCTAGC"},       # لا تشابه
    {"id": "seq_004", "seq": "GACTAGCTAGCTAGGTAGCNGG"},  # تشابه جزئي
]
for test in test_sequences:
    result = checker.check_sequence(test["seq"], test["id"])
    print(f"{test['id']}: {result['classification']} (تشابه: {result['max_similarity']:.2f})")
print("\nقائمة انتظار المراجعة:")
for case in checker.get_review_queue():
    print(f"  {case['seq_id']}: {case['classification']}")
print("\nسجل التدقيق:")
for entry in checker.get_audit_log():
    print(f"  {entry['seq_id']}: {entry['action']}")
```

الدوال المعرفة هنا (__init__, check_sequence, get_audit_log, get_review_queue) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (PathogenCheck) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم

في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة PathogenCheck يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

This educational example demonstrates the basic operation of pathogen_check: screening, classification, sending for review, and recording decisions. However, it does not cover all complexities, nor does it constitute an operational system.

هذا المثال التعليمي يوضح التشغيل الأساسي لـ pathogen_check: الفحص، التصنيف، الإرسال للمراجعة، وتسجيل القرارات. لكنه لا يغطي جميع التعقيدات، ولا يشكل نظاماً تشغيلياً.

6. Updating the Reference and Its Impact

6. تحديث المرجع وتأثيره

In real systems, reference databases change constantly. pathogen_check allows simulating reference updates and observing their impact on system performance. Adding new sequences may reveal sequences that previously appeared safe, or may change the classification of other sequences.

في الأنظمة الحقيقية، تتغير قواعد البيانات المرجعية باستمرار. pathogen_check يسمح بمحاكاة تحديث المرجع ومراقبة تأثيره على أداء النظام. إضافة تسلسلات جديدة قد يكشف تسلسلات كانت تبدو آمنة سابقاً، أو قد يغير تصنيف تسلسلات أخرى.

```
# محاكاة تحديث المرجع
def update_reference(checker, new_sequences):
  """تحديث قاعدة البيانات المرجعية."""
  for seq_id, seq in new_sequences.items():
    checker.reference_db[seq_id] = seq
  # إعادة فحص التسلسلات المعلقة (محاكاة)
  print("تم تحديث المرجع. يجب إعادة فحص التسلسلات المتأثرة.")
```

الدوال المعرفة هنا (update_reference) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص،

وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

This educational simulation shows that the system is not static; it must be updated and evaluated regularly.

7. Not Including Dangerous Lists

هذه المحاكاة التعليمية توضح أن النظام ليس ثابتاً؛ يجب تحديثه وتقييمه بانتظام.

It is important to emphasize that pathogen_check does not contain operational lists of pathogens or toxins. All data used is synthetic and neutral. This is a deliberate design decision: the goal is to teach principles of defensive screening, not to provide a tool for identifying real threats. In real systems, approved and updated databases are used, with appropriate security mechanisms. The educational model does not replace that.

7. عدم تضمين قوائم خطرة

من المهم التأكيد على أن pathogen_check لا يحتوي على قوائم تشغيلية لمسببات الأمراض أو السموم. جميع البيانات المستخدمة هي اصطناعية ومحايدة. هذا قرار تصميم متعمد: الهدف هو تعليم مبادئ الفحص الدفاعي، وليس توفير أداة لتحديد التهديدات الحقيقية. في الأنظمة الحقيقية، تُستخدم قواعد بيانات معتمدة ومحدّثة، مع آليات أمنية مناسبة. النموذج التعليمي لا يغني عن ذلك.

pathogen_check: ما لا يفعله

- لا يحتوي على تسلسلات حقيقية لمسببات الأمراض.

- لا يقدم إرشادات تشغيلية للتعامل مع التهديدات الحقيقية.

- لا يحل محل الأنظمة الأمنية المعتمدة.

- لا يصف كيفية تجاوز الفحص.

Chapter Summary

خلاصة الفصل

pathogen_check is an educational module for defensive sequence screening using neutral synthetic data. It aims to teach screening principles: classification, error measurement (false positives and false negatives), handling uncertainty through human review, and documenting decisions in an audit log. The module emphasizes the importance of updating reference databases and evaluating

system performance. It does not contain dangerous lists and is designed for education only. The next chapter will move to the off_target_scan module for off-target analysis.

pathogen_check هي وحدة تعليمية لفحص التسلسلات الدفاعي باستخدام بيانات اصطناعية محايدة. تهدف إلى تعليم مبادئ الفحص: كيفية التصنيف، قياس الأخطاء (الإيجابيات الكاذبة والسلبيات الكاذبة)، التعامل مع حالات عدم اليقين من خلال المراجعة البشرية، وتوثيق القرارات في سجل تدقيق. تؤكد الوحدة على أهمية تحديث قواعد البيانات المرجعية وتقييم أداء النظام. لا تحتوي على قوائم خطرة، وهي مصممة للتعليم فقط. الفصل القادم سينتقل إلى وحدة off_target_scan لتحليل المواقع غير المستهدفة.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "pathogen_check" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "pathogen_check" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "pathogen_check can detect all pathogens." This is incorrect; the module is educational and uses synthetic data. The correct statement is: "pathogen_check is an educational model illustrating principles of defensive screening, using synthetic data to teach classification, error measurement, and human review, but it is not an operational tool for detecting real pathogens." A simple test: Practical check: "Does this module contain real pathogen data?".

مثال على التجاوز: القول إن "pathogen_check يمكنه اكتشاف جميع مسببات الأمراض". هذا غير صحيح؛ الوحدة تعليمية وتستخدم بيانات اصطناعية. الصحيح هو: "pathogen_check هو نموذج تعليمي يوضح مبادئ الفحص الدفاعي، ويستخدم بيانات اصطناعية لتعليم التصنيف، وقياس الأخطاء، والمراجعة البشرية، لكنه ليس أداة تشغيلية للكشف عن مسببات الأمراض الحقيقية." اختبار بسيط: فحص تطبيقي: "هل هذه الوحدة تحتوي على بيانات حقيقية لمسببات الأمراض؟".

Example: A pathogen_check result classifying a sequence as "potential pathogen" is a computational representation. Claiming that "this sequence is a real pathogen" is a biological claim not proven by the educational model. The classification is based on similarity to synthetic data, not on biological evidence. Practical test: before inferring danger from an educational model, verify the context and additional evidence.

مثال: نتيجة pathogen_check تصنف تسلسلاً على أنه "ممرض محتمل" هو تمثيل حسابي. الادعاء بأن "هذا التسلسل هو مسبب مرض حقيقي" هو ادعاء بيولوجي لا يثبته النموذج التعليمي. التصنيف يعتمد على تشابه مع بيانات اصطناعية، وليس على أدلة بيولوجية. اختبار عملي: قبل أن تستنتج خطورة من نموذج تعليمي، تحقق من السياق والأدلة الإضافية.

مثال تطبيقي على pathogen_check:

· المدخلات: تسلسل استعلام، قاعدة بيانات اصطناعية، عتبة تشابه، سياسة تصنيف.

· الافتراضات: أن البيانات الاصطناعية تمثل سيناريوهات تعليمية، وأن عتبة التشابه المختارة معقولة، وأن سياسة التصنيف واضحة.

· المخرجات: تصنيف (آمن، ممرض محتمل)، إجراء (موافقة، مراجعة)، سجل تدقيق.

· القيود: لا تحتوي على بيانات حقيقية، والسياسات مبسطة، والاختبارات تغطي سيناريوهات محدودة، ولا تقدم إرشادات تشغيلية.

Applied example of pathogen_check:

اختبار: قبل استخدام pathogen_check، تأكد من فهمك لحدودها، وأنك لا تستخدمها في سياق يتطلب بيانات حقيقية.

· Inputs: query sequence, synthetic database, similarity threshold, classification policy.

· Assumptions: that synthetic data represents educational scenarios, that the chosen threshold is reasonable, and that the classification policy is clear.

· Outputs: classification (safe, potential pathogen), action (approve, review), audit log.

· Limitations: it does not contain real data, policies are simplified, tests cover limited scenarios, and it does not provide operational guidance.

Test: before using pathogen_check, ensure you understand its limitations, and that the workflow is not using it in a context requiring real data.

عند استخدام pathogen_check، يجب تسجيل:

· المصدر: البيانات الاصطناعية المستخدمة، وإصدارها.

· الإصدار: إصدار الوحدة، وإصدار أي مكتبات مستخدمة.

· المعلمات: عتبة التشابه، سياسة التصنيف، أي معاملات أخرى.

· التحويلات: أي تعديلات على البيانات أو الكود قبل الاستخدام.

When using pathogen_check, record:

· Provenance: the synthetic data used, and its version.

· Version: the module version, and any library versions used.

· Parameters: the similarity threshold, classification policy, and any other parameters.

· Transformations: any modifications to the data or code before use.

pathogen_check نفسه هو مثال على استخدام اختبارات اصطناعية. قبل تطبيق أي نظام فحص على بيانات حقيقية، اختبره على بيانات اصطناعية. استخدم pathogen_check كمنصة لاختبار سياسات مختلفة، وفهم كيفية عمل النظام في سيناريوهات مختلفة. إذا نجحت الاختبارات على البيانات الاصطناعية، فأنت واثق من أن منطق النظام يعمل كما هو متوقع. وعند الانتقال إلى بيانات حقيقية، تذكر أن البيانات الاصطناعية لا تغطي كل التعقيدات الواقعية، وقد تتطلب تعديلات.

pathogen_check itself is an example of using synthetic tests. Before applying any screening system to real data, test it on synthetic data. Use pathogen_check as a platform to test different policies, and understand how the system works in different scenarios. If the tests succeed on synthetic data, the workflow is confident that the system logic works as expected. When moving to real data, remember that synthetic data does not cover all real-world complexities, and may require adjustments.

Chapter 39 — off_target_scan

الفصل

الفكرة المركزية: مسح تشابه محايد، ranking، حدود التنبؤ، human review.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Defensive similarity screening should remain traceable: what synthetic sequence was used, what distance criterion was applied, what search limits were used, and what result triggered review? These questions make the tool testable without turning it into an experimental design system.

الفحص الدفاعي لمواقع التشابه يجب أن يظل قابلاً للتتبع: ما التسلسل الاصطناعي المستخدم، وما معيار المسافة، وما حدود البحث، وما الذي اعتُبر نتيجة تستوجب المراجعة؟ هذه الأسئلة تجعل الأداة قابلة للاختبار دون أن تتحول إلى نظام تصميم تجريبي.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Educational Scanning for Off-Target Sites

1. المسح التعليمي للمواقع غير المستهدفة

off_target_scan is an educational module that compares a guide RNA sequence with a reference genome (synthetic data) to generate a list of sites with high similarity. This scan is neutral: it does not assume that similarity means activity, but generates candidates for review. This approach reflects sound scientific practice, where similarity is used as an initial filtering tool, not as definitive evidence.

off_target_scan هي وحدة تعليمية تقارن تسلسل guide RNA مع جينوم مرجعي (بيانات اصطناعية) لتوليد قائمة بالمواقع ذات التشابه العالي. هذا المسح محايد: لا يفترض أن التشابه يعني النشاط، بل يولد مرشحين للمراجعة. هذا النهج يعكس الممارسة العلمية السليمة، حيث يُستخدم التشابه كأداة للتصفية الأولية، وليس كدليل نهائي.

مبدأ المسح التعليمي:

1. إدخال تسلسل guide RNA.

2. مقارنته مع جينوم اصطناعي (محاكاة).

3. حساب التشابه لكل موقع.

4. ترتيب المواقع حسب التشابه.

5. إخراج قائمة مرشحين للمراجعة البشرية.

2. Understanding Candidate Generation

Candidate generation is a purely computational process: it involves no biological judgment. The goal is to narrow the search from the entire genome to a small set of sites worthy of further attention. In off_target_scan, textual similarity (the number of matches between the guide RNA and the site) is used as the criterion for generating candidates. The system may generate many or few candidates depending on the chosen threshold. A higher threshold produces fewer candidates

(higher specificity), but may miss important sites (lower sensitivity). A lower threshold produces more candidates (higher sensitivity), but may include many false positives.

2. فهم توليد المرشحين (Candidate Generation)

توليد المرشحين هو عملية حسابية بحتة: لا تتضمن أي حكم بيولوجي. الهدف هو تضييق نطاق البحث من الجينوم الكامل إلى مجموعة صغيرة من المواقع التي تستحق المزيد من الاهتمام. في off_target_scan، يُستخدم التشابه النصي (عدد التطابقات بين guide RNA والموقع) كمعيار لتوليد المرشحين. قد يخرج النظام مرشحين كثيرين أو قليلين بحسب العتبة المختارة. العتبة الأعلى تنتج عدداً أقل من المرشحين (نوعية أعلى)، لكنها قد تفوت مواقع مهمة (حساسية أقل). العتبة الأدنى تنتج عدداً أكبر من المرشحين (حساسية أعلى)، لكنها قد تتضمن الكثير من الإيجابيات الكاذبة.

```
# مثال تعليمي: توليد مرشحين لمواقع غير مستهدفة
def generate_candidates(guide_rna, genome, threshold=0.8):
    """
```

توليد مرشحين للمواقع غير المستهدفة بناءً على التشابه.

للاستخدام التعليمي فقط (بيانات اصطناعية).

```
    """
    candidates = []
    for position, sequence in genome.items():
        matches = sum(1 for a, b in zip(guide_rna, sequence) if a == b)
        similarity = matches / len(guide_rna) if guide_rna else 0
        if similarity >= threshold:
            candidates.append({
                "position": position,
                "sequence": sequence,
                "similarity": similarity,
                "mismatches": len(guide_rna) - matches
            })
    # ترتيب تنازلي حسب التشابه
    candidates.sort(key=lambda x: x["similarity"], reverse=True)
    return candidates
```

الدوال المعرفة هنا (generate_candidates) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة.

القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational example shows that candidate generation is just a first step, not the end of the analysis.

3. Limits of the Computational Result: Not an Experimental Recommendation

هذا المثال التعليمي يوضح أن توليد المرشحين هو مجرد خطوة أولى، وليس نهاية التحليل.

A common mistake is treating the candidate list as if it were an experimental recommendation. In reality, the candidate list is just a starting point for further investigation. Computational similarity does not equal biological activity. A site may be 90% similar to the guide RNA, but not be cut in the cell due to factors such as PAM availability, chromatin packaging, or DNA modifications. Therefore, the candidate list should be presented as a decision-support tool, not as a substitute for laboratory experiments or human review.

3. حدود النتيجة الحسابية: ليست توصية تجريبية

من الأخطاء الشائعة التعامل مع قائمة المرشحين كما لو كانت توصية تجريبية. في الواقع، قائمة المرشحين هي مجرد نقاط انطلاق لمزيد من التحقيق. التشابه الحسابي لا يساوي النشاط البيولوجي. قد يكون موقع ما متشابهاً بنسبة 90% مع guide RNA، لكنه لا يُقطع في الخلية بسبب عوامل مثل توفر PAM، أو تغليف الكروماتين، أو تعديلات الـ DNA. لذلك، يجب أن تُعرض قائمة المرشحين كأداة لدعم القرار، وليس كبديل عن التجارب المعملية أو المراجعة البشرية.

ما تعنيه قائمة المرشحين:

- مواقع تستحق الاهتمام.

- مؤشرات أولية، وليست حقائق.

- تحتاج إلى تحقق إضافي.

- جزء من عملية اتخاذ القرار، وليست القرار نفسه.

4. Documenting Thresholds and References

In off_target_scan, as in any screening system, documenting thresholds and references is the foundation of reproducibility. The threshold used (e.g., 0.8) affects the candidate list. The reference used (e.g., the genome version) affects the specific sites identified. If these elements are not documented, no one can reinterpret the result or compare it with other studies. Therefore, the following must be recorded:

4. توثيق العتبات والمراجع

في off_target_scan، كما في أي نظام فحص، توثيق العتبات والمراجع هو أساس قابلية إعادة الإنتاج. العتبة المستخدمة (مثل 0.8) تؤثر على قائمة المرشحين. المرجع المستخدم (مثل إصدار الجينوم) يؤثر على المواقع المحددة. إذا لم توثق هذه العناصر، فلن يستطيع أحد إعادة تفسير النتيجة أو مقارنتها مع دراسات أخرى. لذلك، يجب تسجيل:

· عتبة التشابه المستخدمة.

· The similarity threshold used.

· إصدار الجينوم المرجعي (وتاريخه).

· The reference genome version (and its date).

· تسلسل guide RNA المستخدم.

· The guide RNA sequence used.

· أي معاملات أخرى (مثل طول المطابقة، عدد الاختلافات المسموح بها).

· Any other parameters (e.g., match length, number of allowed mismatches).

```
# مثال: توثيق المعلمات في سجل
parameters = {
    "guide_rna": "GACTAGCTAGCTAGCTAGC",
    "threshold": 0.8,
    "reference_version": "synthetic_v1.0",
    "reference_date": "2025-01-15",
    "allowed_mismatches": 2
}
```

This documentation allows re-analysis and interpretation of results in their correct context.

5. Avoiding Turning Similarity Results into Experimental Recommendations

هذا التوثيق يسمح بإعادة التحليل وتفسير النتائج في سياقها الصحيح.

In high-consequence systems, the default path should be conservative. The absence of authorization, trusted reference, or audit trail should not become "implicit approval." In off_target_scan, this means the candidate list is not used directly for experimental recommendations, but is presented for human review. The human reviewer evaluates the context, considers additional factors, and makes the decision. This ensures that decisions are not fully automated, but subject to human accountability.

5. تجنب تحويل نتائج التشابه إلى توصيات تجريبية

في الأنظمة ذات العواقب العالية، يجب أن يكون المسار الافتراضي محافظاً. غياب الإذن أو المرجع الموثوق أو سجل التدقيق لا ينبغي أن يتحول إلى "سماح ضمني". في off_target_scan، هذا يعني أن قائمة المرشحين لا تُستخدم مباشرة لتوصيات تجريبية، بل تُعرض للمراجعة البشرية. المراجع البشري يقيّم السياق، ويأخذ في الاعتبار العوامل الإضافية، ويتخذ القرار. هذا يضمن أن القرارات ليست آلية تماماً، بل تخضع للمساءلة البشرية.

المسار الدفاعي في off_target_scan:

1. توليد مرشحين.

2. عرضهم للمراجعة البشرية.

3. المراجع يقرر الخطوة التالية (اختبار معملي، تعديل الدليل، استبعاد).

4. تسجيل القرار في سجل التدقيق.

6. Practical Application: Running off_target_scan

Let us apply off_target_scan in a simple educational example. We will use a small synthetic genome, a known guide RNA, and generate a candidate list for off-target sites.

6. التطبيق العملي: تشغيل off_target_scan

دعنا نطبق off_target_scan في مثال تعليمي بسيط. سنستخدم جينوماً اصطناعياً صغيراً، وguide RNA معروف، ونولد قائمة مرشحين للمواقع غير المستهدفة.

```
# محاكاة تعليمية لـ off_target_scan
class OffTargetScan:
    def __init__(self, reference_genome, guide_rna, similarity_threshold=0.8):
        self.reference_genome = reference_genome
        self.guide_rna = guide_rna
        self.similarity_threshold = similarity_threshold
        self.audit_log = []
def scan(self):
        """مسح الجينوم لتوليد مرشحين للمواقع غير المستهدفة."""
        candidates = []
        for position, sequence in self.reference_genome.items():
            matches = sum(1 for a, b in zip(self.guide_rna, sequence) if a == b)
            similarity = matches / len(self.guide_rna) if self.guide_rna else 0
            if similarity >= self.similarity_threshold:
                candidates.append({
                    "position": position,
                    "sequence": sequence,
```

```
                "similarity": similarity,
                "mismatches": len(self.guide_rna) - matches
            })
        # تسجيل في سجل التدقيق
        self.audit_log.append({
            "action": "scan",
            "guide_rna": self.guide_rna,
            "threshold": self.similarity_threshold,
            "candidates_count": len(candidates),
            "timestamp": "2025-01-15T12:00:00"
        })
        return candidates
    def get_audit_log(self):
        return self.audit_log
# بيانات اصطناعية (جينوم صغير)
synthetic_genome = {
    "chr1:100-120": "GACTAGCTAGCTAGCTAGC",   # هدف مثالي (تطابق تام)
    "chr1:200-220": "GACTAGCTAGCTAGCTATC",   # اختلاف واحد
    "chr1:300-320": "GACTAGCTAGCTAGGTAGC",   # اختلافان
    "chr2:100-120": "GACTAGCTAGCTAGCTAGC",   # تطابق تام (موقع آخر)
    "chrX:500-520": "ATGCGTACGTAGCTAGC",     # لا تشابه
}
guide = "GACTAGCTAGCTAGCTAGC"
scanner = OffTargetScan(synthetic_genome, guide, similarity_threshold=0.8)
candidates = scanner.scan()
print("نتائج مسح المواقع غير المستهدفة (محاكاة تعليمية):")
for c in candidates:
    print(f"  {c['position']}: تشابه {c['similarity']:.2f}, اختلافات: {c['mismatches']}")
print(f"\nعدد المرشحين: {len(candidates)}")
print("سجل التدقيق:", scanner.get_audit_log())
```

الدوال المعرفة هنا (__init__, scan, get_audit_log) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (OffTargetScan) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل

الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة OffTargetScan يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

This educational example demonstrates the basic operation of off_target_scan: scanning, generating candidates, and ranking by similarity. However, it does not cover all complexities, nor does it constitute an operational system.

هذا المثال التعليمي يوضح التشغيل الأساسي لـ off_target_scan: المسح، توليد المرشحين، والترتيب حسب التشابه. لكنه لا يغطي جميع التعقيدات، ولا يشكل نظاماً تشغيلياً.

7. Limits of the Educational Scan

7. حدود المسح التعليمي

off_target_scan is an educational model with clear limits:

off_target_scan هو نموذج تعليمي، وله حدود واضحة:

· لا يحتوي على جينومات حقيقية أو بيانات حساسة.

· لا يستخدم خوارزميات تحسين أو تنبؤ متقدمة.

· It does not contain real genomes or sensitive data.

· لا يأخذ في الاعتبار العوامل البيولوجية (مثل PAM، الكروماتين).

· It does not use advanced optimization or prediction algorithms.

· لا يقدم توصيات تجريبية.

· It does not account for biological factors (such as PAM, chromatin).

· المخرجات مصممة للمراجعة البشرية، وليس للأتمتة.

· It does not provide experimental recommendations.

· Outputs are designed for human review, not for automation.

Chapter Summary

خلاصة الفصل

off_target_scan is an educational module for neutral similarity scanning and generating candidates for off-target sites. It relies on comparing a guide RNA with a synthetic genome, calculating similarity, and ranking results. The candidate list is not an experimental recommendation, but a tool for human review. Choosing a similarity threshold affects sensitivity and specificity. Documenting thresholds and references is the foundation of reproducibility. Outputs are presented for human review, not for blind automation. The next chapter will move to gene_drive_lock as a model for software constraints and authorization.

off_target_scan هي وحدة تعليمية لمسح التشابه المحايد وتوليد مرشحين للمواقع غير المستهدفة. تعتمد على مقارنة guide RNA مع جينوم اصطناعي، وحساب التشابه، وترتيب النتائج. قائمة المرشحين ليست توصية تجريبية، بل أداة للمراجعة البشرية. اختيار عتبة التشابه يؤثر على الحساسية والنوعية. توثيق العتبات والمراجع هو أساس إعادة الإنتاج. المخرجات تُقدم للمراجعة البشرية، وليس للأتمتة العمياء. الفصل القادم سينتقل إلى gene_drive_lock كنموذج للقيود البرمجية والصلاحيات.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "off_target_scan" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "off_target_scan" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "off_target_scan can predict all off-target sites with accuracy." This is incorrect; the module generates candidates based on similarity, but it does not predict actual activity. The correct statement is: "off_target_scan is an educational model for generating candidates for off-target sites based on similarity, but it is not a prediction tool, and results require human review and further verification." A simple test: Practical check: "Does this list represent sites that will actually be cut, or just candidates for review?".

مثال على التجاوز: القول إن "off_target_scan يمكنه التنبؤ بجميع المواقع غير المستهدفة بدقة". هذا غير صحيح؛ الوحدة تولد مرشحين بناءً على التشابه، لكنها لا تتنبأ بالنشاط الفعلي. الصحيح هو: "off_target_scan هو نموذج تعليمي لتوليد مرشحين للمواقع غير المستهدفة بناءً على التشابه، لكنه ليس أداة تنبؤ، وتحتاج النتائج إلى مراجعة بشرية وتحقق إضافي." اختبار بسيط: فحص تطبيقي: "هل هذه القائمة تمثل مواقع ستُقطع فعلاً، أم مجرد مرشحين للمراجعة؟".

Example: A candidate list containing sites with 85% similarity is a computational representation. Claiming that "these sites will be cut in the cell" is a biological claim not proven by similarity alone. Similarity prioritizes sites, but actual activity depends on cellular factors. Practical test: before inferring activity from a candidate list, verify additional context (such as PAM presence, accessibility).

مثال: قائمة مرشحين تحتوي على مواقع بتشابه 85% هي تمثيل حسابي. الادعاء بأن "هذه المواقع ستُقطع في الخلية" هو ادعاء بيولوجي لا يثبته التشابه وحده. التشابه يرشح المواقع، لكن النشاط الفعلي يعتمد على عوامل خلوية.

اختبار عملي: قبل أن تستنتج نشاطاً من قائمة مرشحين، تحقق من السياق الإضافي (مثل وجود PAM، إمكانية الوصول).

مثال تطبيقي على off_target_scan:

· المدخلات: تسلسل guide RNA، جينوم اصطناعي، عتبة تشابه.

· الافتراضات: أن التشابه النصي مؤشر أولي معقول، وأن الجينوم الاصطناعي يمثل سيناريو تعليمي، وأن العتبة المختارة مناسبة.

· المخرجات: قائمة مرشحين مرتبة حسب التشابه، مع معلومات عن عدد الاختلافات.

· القيود: لا يأخذ في الاعتبار العوامل البيولوجية، ولا يتنبأ بالنشاط الفعلي، ولا يقدم توصيات تجريبية.

Applied example of off_target_scan:

اختبار: قبل استخدام off_target_scan، تأكد من فهمك لحدودها، وأنك لا تستخدمها كأساس لقرارات تجريبية.

· Inputs: guide RNA sequence, synthetic genome, similarity threshold.

· Assumptions: that textual similarity is a reasonable initial indicator, that the synthetic genome represents an educational scenario, and that the chosen threshold is appropriate.

· Outputs: a list of candidates ranked by similarity, with information on the number of mismatches.

· Limitations: it does not account for biological factors, does not predict actual activity, and does not provide experimental recommendations.

Test: before using off_target_scan, ensure you understand its limitations, and that the workflow is not using it as a basis for experimental decisions.

عند استخدام off_target_scan، يجب تسجيل:

· المصدر: الجينوم الاصطناعي المستخدم، وإصداره.

· المعلمات: عتبة التشابه، تسلسل guide RNA، أي معاملات أخرى.

When using off_target_scan, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة تشغيل المسح والحصول على نفس قائمة المرشحين؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the synthetic genome used, and its version.

· Parameters: the similarity threshold, guide RNA sequence, and any other parameters.

Test: Practical check: "If I gave this record to a colleague a year later, could they rerun the scan and obtain the same candidate list?" If the answer is no, the documentation is insufficient.

off_target_scan نفسه هو مثال على استخدام اختبارات اصطناعية. قبل تطبيق أي نظام مسح على بيانات حقيقية، اختبره على بيانات اصطناعية. استخدم off_target_scan كمنصة لفهم كيفية تأثير العتبات والمعلمات على قائمة المرشحين. إذا نجحت الاختبارات على البيانات الاصطناعية، فأنت واثق من أن منطق النظام يعمل كما هو متوقع. وعند الانتقال إلى بيانات حقيقية، تذكر أن البيانات الاصطناعية لا تغطي كل التعقيدات الواقعية، وقد تتطلب تعديلات.

off_target_scan itself is an example of using synthetic tests. Before applying any screening system to real data, test it on synthetic data. Use off_target_scan as a platform to understand how thresholds and parameters affect the candidate list. If the tests succeed on synthetic data, the workflow is confident that the system logic works as expected. When moving to real data, remember that synthetic data does not cover all real-world complexities, and may require adjustments.

Chapter 40 — gene_drive_lock

الفصل

الفكرة المركزية: authorization, policy gates, fail-closed, audit log.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

قفل gene-drive في الحزمة التعليمية يمثل مبدأاً أمنياً: عند غياب شروط السلامة أو وجود حالة غير معروفة، يكون التوقف هو الناتج المقصود. هذه ليست علامة فشل برمجي؛ إنها سياسة قرار.

The educational gene-drive lock represents a security principle: when safety conditions are absent or the state is unknown, stopping is the intended output. This is not a programming failure; it is a decision policy.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is gene_drive_lock؟

1. ما هو gene_drive_lock؟

gene_drive_lock is an educational module representing a model for software constraints and authorization in sensitive systems. The basic idea is that any sensitive operation (such as executing a gene drive, releasing a modified organism, or a high-risk experiment) must pass through an authorization gate. This gate verifies the presence of explicit approval, fulfillment of required conditions, and records the decision in an audit log. If any of these elements are absent, execution stops or escalates to human review. The module does not provide instructions for building a gene drive, but a model for security governance.

gene_drive_lock هي وحدة تعليمية تمثل نموذجاً للقيود البرمجية والصلاحيات في الأنظمة الحساسة. الفكرة الأساسية هي أن أي عملية حساسة (مثل تنفيذ محرك جيني، أو إطلاق كائن معدل، أو تجربة عالية المخاطر) يجب أن تمر عبر بوابة صلاحيات (authorization gate). هذه البوابة تتحقق من وجود موافقة صريحة، واستيفاء الشروط المطلوبة، وتسجيل القرار في سجل تدقيق. إذا غاب أي من هذه العناصر، يتوقف التنفيذ أو يُصعد للمراجعة البشرية.

الوحدة لا تقدم تعليمات لبناء محرك جيني، بل نموذجاً للحوكمة الأمنية.

مبدأ gene_drive_lock:

لا شيء حساس يحدث دون:

1. موافقة صريحة موثقة.

2. تحقق من جميع الشروط المطلوبة.

3. تسجيل القرار في سجل تدقيق.

4. إمكانية التوقف أو التصعيد للمراجعة البشرية.

2. فهم سلوك fail-closed

2. Understanding Fail-Closed Behavior

In traditional systems, the default behavior may be "allow unless explicitly denied" (fail-open). In critical systems, the default behavior must be "deny unless explicitly allowed" (fail-closed). This means that the absence of authorization, trusted reference, or audit log leads to halting, not continuation. This principle ensures that any error or vulnerability does not lead to unauthorized execution. gene_drive_lock applies this principle strictly: the gate cannot be bypassed without explicit approval and condition verification.

في الأنظمة التقليدية، قد يكون السلوك الافتراضي هو "السماح ما لم يُمنع صراحة" (fail-open). في الأنظمة الحرجة، يجب أن يكون السلوك الافتراضي هو "المنع ما لم يُسمح صراحة" (fail-closed). هذا يعني أن غياب الإذن أو المرجع الموثوق أو سجل التدقيق يؤدي إلى التوقف، وليس إلى الاستمرار. هذا المبدأ يضمن أن أي خطأ أو ثغرة لا تؤدي إلى تنفيذ غير مصرح به. gene_drive_lock يطبق هذا المبدأ بشكل صارم: لا يمكن تجاوز البوابة بدون موافقة صريحة وتحقق من الشروط.

```
# مثال تعليمي: تطبيق fail-closed
def gene_drive_lock(request, approval, conditions_met, audit_log):
    """
```

بوابة صلاحيات تطبق مبدأ fail-closed.

```
    """
    # التحقق من الموافقة الصريحة
    if not approval:
        audit_log.append("فشل: لا توجد موافقة صريحة")
        return {"status": "denied", "reason": "no explicit approval"}
# التحقق من الشروط
    if not conditions_met:
        audit_log.append("فشل: الشروط غير مستوفاة")
        return {"status": "pending_review", "reason": "conditions not met"}
# تنفيذ الإجراء (محاكاة)
    return {"status": "executed", "reason": "success"}
```

This educational example shows that the gate only allows execution after verifying approval and conditions.

3. Separation of Duties and Responsibilities

هذا المثال التعليمي يوضح أن البوابة لا تسمح بالتنفيذ إلا بعد التحقق من الموافقة والشروط.

In sensitive systems, duties must be separated so that no single person has the ability to execute a sensitive action alone. This reduces the risk of misuse or errors. In gene_drive_lock, duties are separated into several roles:

3. فصل الصلاحيات والمسؤوليات

في الأنظمة الحساسة، يجب فصل الصلاحيات بحيث لا يمتلك شخص واحد القدرة على تنفيذ إجراء حساس بمفرده. هذا يقلل من خطر سوء الاستخدام أو الأخطاء. في gene_drive_lock، يتم فصل الصلاحيات إلى عدة أدوار:

· مقدم الطلب (Requester): يطلب الإجراء ويقدم الأساس المنطقي.

· المراجع (Reviewer): يفحص الطلب، ويقرر الموافقة أو الرفض، ويسجل قراره.

· Requester: requests the action and provides the rationale.

· المدقق (Auditor): يراجع سجلات التدقيق للتحقق من الامتثال.

· Reviewer: examines the request, decides to approve or deny, and records their decision.

· المنفذ (Executor): ينفذ الإجراء بعد الحصول على الموافقات.

· Auditor: reviews audit logs to verify compliance.

· Executor: executes the action after obtaining approvals.

مبدأ فصل الصلاحيات:

لا يمكن لنفس الشخص أن يطلب ويراجع وينفذ إجراءً حساساً.

هذا يقلل من خطر التجاوزات والأخطاء.

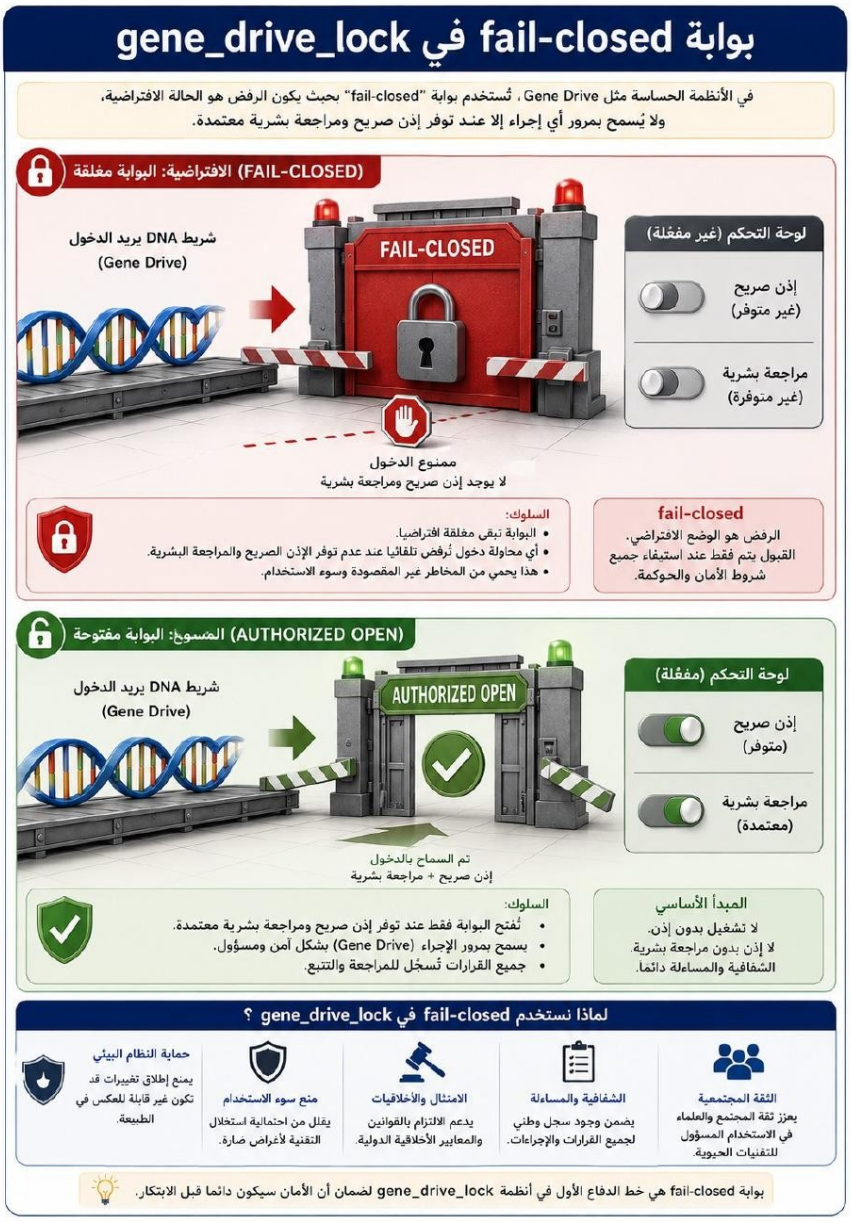
4. Audit Logs and Explicit Approval

Audit logs are the backbone of accountability in gene_drive_lock. Every decision is recorded with its details: who requested, who approved, when, why, and what conditions were met. This allows tracking any decision, identifying errors, and improving processes. Explicit approval means that the approval is recorded and documented, not implicit or verbal. In gene_drive_lock, no sensitive action is executed without explicit approval recorded in the audit log.

4. سجلات التدقيق والموافقة الصريحة

سجل التدقيق هو العمود الفقري للمساءلة في gene_drive_lock. كل قرار يتم تسجيله بتفاصيله: من طلب، ومن وافق، ومتى، ولماذا، وما هي الشروط التي تحققت. هذا يسمح بتتبع أي قرار، وتحديد الأخطاء، وتحسين العمليات.

الموافقة الصريحة تعني أن الموافقة مسجلة وموثقة، وليست ضمنية أو شفهية. في gene_drive_lock، لا يتم تنفيذ أي إجراء حساس دون موافقة صريحة مسجلة في سجل التدقيق.



Infographic 21 / 21 إنفوجرافيك


```
مثال: تسجيل موافقة في سجل التدقيق#
def record_approval(audit_log, request_id, reviewer_id, decision, rationale):
    """
```

تسجيل قرار الموافقة في سجل التدقيق.

```
"""
entry = {
    "request_id": request_id,
    "reviewer_id": reviewer_id,
    "decision": decision,
    "rationale": rationale,
    "timestamp": "2025-01-15T12:00:00",
    "conditions_met": ["condition_1", "condition_2"]
}
audit_log.append(entry)
return entry
```

الدوال المعرفة هنا (record_approval) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

This educational example shows that approval is recorded with its details, ensuring auditability.

5. Applying the Fail-Closed Principle in Practice

هذا المثال التعليمي يوضح أن الموافقة تُسجل بتفاصيلها، مما يضمن قابلية التدقيق.

In practice, applying fail-closed means the system must be designed so that the gate cannot be bypassed in any way. This requires:

5. تطبيق مبدأ fail-closed في الممارسة

في الممارسة، تطبيق fail-closed يعني أن النظام يجب أن يكون مصمماً بحيث لا يمكن تجاوز البوابة بأي طريقة. هذا يتطلب:

· مصادقة قوية: للتحقق من هوية المستخدمين.

· تسجيل شامل: لكل محاولة وصول أو تنفيذ.

· Strong authentication: to verify user identity.

· مراجعة دورية: لسجلات التدقيق للكشف عن أي محاولات غير مصرح بها.

· Comprehensive logging: for every access or execution attempt.

· تحديث السياسات: بانتظام لمواكبة التهديدات الجديدة.

· Periodic review: of audit logs to detect any unauthorized attempts.

· تدريب المستخدمين: على أهمية الأمن والمساءلة.

· Policy updates: regularly to keep pace with new threats.

· User training: on the importance of security and accountability.

تطبيق fail-closed:

- لا يتم تنفيذ أي إجراء حساس دون موافقة.

- كل محاولة تنفيذ تُسجل.

- المحاولات غير المصرح بها تُمنع وتُبلغ.

- يتم مراجعة السجلات بانتظام.

- يتم تحديث السياسات بناءً على الدروس المستفادة.

6. Practical Application: Running gene_drive_lock

Let us apply gene_drive_lock in a simple educational example. We will simulate a request to execute a sensitive action, pass it through the authorization gate, and record the decision.

6. التطبيق العملي: تشغيل gene_drive_lock

دعنا نطبق gene_drive_lock في مثال تعليمي بسيط. سنحاكي طلب تنفيذ إجراء حساس، ونمرره عبر بوابة الصلاحيات، ونسجل القرار.

```
def request_action(self, request_id, requester, action, rationale):
    """تقديم طلب لتنفيذ إجراء حساس."""
    entry = {
        "request_id": request_id,
        "requester": requester,
        "action": action,
        "rationale": rationale,
        "status": "pending",
        "timestamp": "2025-01-15T12:00:00"
    }
    self.audit_log.append(entry)
    return entry
# محاكاة تعليمية لـ gene_drive_lock
class GeneDriveLock:
  def __init__(self):
    self.audit_log = []
```

محاكاة قاعدة بيانات الموافقات # {} = self.approvals

def review_request(self, request_id, reviewer_id, decision, rationale):

"""مراجعة طلب واتخاذ قرار."""

if decision not in ["approved", "denied"]:
return {"error": "decision must be approved or denied"}

تحديث الطلب في سجل التدقيق #

for entry in self.audit_log:
if entry["request_id"] == request_id:

```
entry["reviewer"] = reviewer_id
entry["decision"] = decision
entry["review_rationale"] = rationale
entry["review_date"] = "2025-01-15T12:30:00"
entry["status"] = decision
break
```

```
# تخزين الموافقة إذا كانت موافقة
    if decision == "approved":
        self.approvals[request_id] = {
```

```
"reviewer": reviewer_id,
"approved_at": "2025-01-15T12:30:00"
}
```

return {"request_id": request_id, "decision": decision}
def execute_action(self, request_id, executor):

"""تنفيذ الإجراء بعد التحقق من الموافقة."""

التحقق من وجود موافقة #

if request_id not in self.approvals:
self.audit_log.append({

```
"request_id": request_id,
"action": "execute_attempt",
"status": "denied",
"reason": "no approval found",
"executor": executor,
"timestamp": "2025-01-15T13:00:00"
```

```
})
        return {"status": "denied", "reason": "no approval found"}
# تنفيذ الإجراء (محاكاة)
    self.audit_log.append({
        "request_id": request_id,
        "action": "execute",
        "status": "executed",
        "executor": executor,
        "timestamp": "2025-01-15T13:00:00"
    })
    return {"status": "executed", "reason": "action completed successfully"}
def get_audit_log(self):
    return self.audit_log
# مثال على الاستخدام
lock = GeneDriveLock()
# 1. تقديم طلب
request = lock.request_action("REQ_001", "researcher_01",
                              "gene_drive_release", "لدراسة تأثير المحرك الجيني")
print(f"طلب: {request['status']}")
# 3. تنفيذ الإجراء
execution = lock.execute_action("REQ_001", "executor_03")
print(f"تنفيذ: {execution['status']}")
# 2. مراجعة الطلب (موافقة)
review = lock.review_request("REQ_001", "reviewer_02", "approved", "تمت المراجعة الأمنية والأخلاقية")
print(f"مراجعة: {review['decision']}")
# عرض سجل التدقيق
print("\nسجل التدقيق:")
for entry in lock.get_audit_log():
    print(f"  {entry}")
```

الدوال المعرفة هنا (__init__, request_action, review_request, execute_action, get_audit_log) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (GeneDriveLock) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي

يجب التحقق منها قبل التحليل. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة GeneDriveLock يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

This educational example demonstrates the basic operation of gene_drive_lock: requesting, reviewing, approving, executing, and recording everything in the audit log.

هذا المثال التعليمي يوضح التشغيل الأساسي لـ gene_drive_lock: تقديم الطلب، المراجعة، الموافقة، التنفيذ، وتسجيل كل شيء في سجل التدقيق.

7. Limits of the Educational Model

7. حدود النموذج التعليمي

gene_drive_lock is an educational model with clear limits:

gene_drive_lock هو نموذج تعليمي، وله حدود واضحة:

· لا يحتوي على أنظمة مصادقة حقيقية أو آليات أمنية تشغيلية.

· لا يقدم تعليمات لبناء أو نشر محرك جيني فعلي.

· السياسات مبسطة ولا تعكس التعقيدات القانونية والأخلاقية للأنظمة الحقيقية.

· لا يحل محل الأنظمة الأمنية المعتمدة أو المراجعة البشرية الفعلية.

· It does not contain real authentication systems or operational security mechanisms.

· الهدف هو تعليم مبادئ الحوكمة الأمنية، وليس تقديم أداة جاهزة.

· It does not provide instructions for building or deploying an actual gene drive.

· Policies are simplified and do not reflect the legal and ethical complexities of real systems.

· It does not replace approved security systems or actual human review.

· The goal is to teach governance principles, not to provide a ready-made tool.

Chapter Summary

خلاصة الفصل

gene_drive_lock is an educational module representing a model for software constraints and authorization in sensitive systems. It applies the fail-closed principle, where no sensitive action is executed without explicit approval and condition verification. Separation of duties and responsibilities reduces the risk of misuse. Audit logs are the backbone of accountability, recording every decision and its details. The module is educational only, and does not provide operational guidance for building gene drives. The next chapter will move to the question: should the package be published on PyPI?

gene_drive_lock هي وحدة تعليمية تمثل نموذجاً للقيود البرمجية والصلاحيات في الأنظمة الحساسة. تطبق مبدأ fail-closed، حيث لا يتم تنفيذ أي إجراء حساس دون موافقة صريحة وتحقق من الشروط. فصل الصلاحيات والمسؤوليات يقلل من خطر سوء الاستخدام. سجلات التدقيق هي العمود الفقري للمساءلة، وتسجل كل قرار وتفاصيله. الوحدة تعليمية فقط، ولا تقدم إرشادات تشغيلية لبناء المحركات الجينية. الفصل القادم سينتقل إلى السؤال: هل تُنشر الحزمة على PyPI؟

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "gene_drive_lock" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "gene_drive_lock" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "gene_drive_lock can prevent all misuse of gene drives." This is incorrect; the model is educational and illustrates governance principles, but it is not a comprehensive solution. The correct statement is: "gene_drive_lock is an educational model illustrating principles of software constraints and authorization, and helps in understanding how to design governance systems for sensitive activities, but it is not an operational tool for preventing misuse." A simple test: Practical check: "Does this model contain sufficient operational security mechanisms to prevent all threats?".

مثال على التجاوز: القول إن "gene_drive_lock يمكنه منع جميع حالات سوء استخدام المحركات الجينية". هذا غير صحيح؛ النموذج تعليمي ويوضح مبادئ الحوكمة، لكنه ليس حلاً شاملاً. الصحيح هو: "gene_drive_lock هو نموذج تعليمي يوضح مبادئ القيود البرمجية والصلاحيات، ويساعد في فهم كيفية تصميم أنظمة حوكمة للأنشطة

الحساسة، لكنه ليس أداة تشغيلية لمنع سوء الاستخدام." اختبار بسيط: فحص تطبيقي: "هل هذا النموذج يحتوي على آليات أمنية تشغيلية كافية لمنع جميع التهديدات؟"ـ

Example: The gene_drive_lock system implementing security policies is a computational representation of those policies. Claiming that "this system guarantees biosafety" is a biological claim not proven by the educational model. The system can apply specific rules, but it cannot evaluate actual biological risks or ethical context. Practical test: before considering a security system "safe," verify the existence of human review and comprehensive policies.

مثال: نظام gene_drive_lock يطبق سياسات أمنية هو تمثيل حسابي لتلك السياسات. الادعاء بأن "هذا النظام يضمن السلامة البيولوجية" هو ادعاء بيولوجي لا يثبته النموذج التعليمي. النظام يمكنه تطبيق قواعد محددة، لكنه لا يستطيع تقييم المخاطر البيولوجية الفعلية أو السياق الأخلاقي. اختبار عملي: قبل أن تعتبر نظاماً أمنياً "آمناً"، تحقق من وجود مراجعة بشرية وسياسات شاملةـ

مثال تطبيقي على gene_drive_lock:

Applied example of gene_drive_lock:

اختبار: قبل الاعتماد على gene_drive_lock، تحقق من آليات المصادقة، وحماية سجل التدقيق، ووجود مراجعة بشرية للحالات المستثناةـ

Test: before relying on gene_drive_lock, verify authentication mechanisms, audit log protection, and the existence of human review for exceptional cases.

عند تصميم أو استخدام نظام قيود مثل gene_drive_lock، يجب تسجيل ما يلي:

When designing or using a constraint system like gene_drive_lock, record the following:

قبل تطبيق أي نظام قيود مثل gene_drive_lock على بيئة حقيقية، اختبره على سيناريوهات اصطناعية. أنشئ حالات اختبار تمثل مواقف مختلفة: طلب صحيح مع موافقات، طلب بدون موافقة، طلب مع شروط غير مستوفاة، محاولة تجاوز. تحقق من أن النظام يتعامل مع كل حالة بشكل صحيح وفقاً للسياسات المحددة، وأن سجل التدقيق يسجل كل قرار. إذا نجح الاختبار، فأنت واثق من أن النظام يعمل كما هو متوقع، وأي حالة مع بيئة حقيقية تعكس تعقيدات العالم الحقيقي التي قد تتطلب مراجعة بشريةـ

Before applying any constraint system like gene_drive_lock to a real environment, test it on synthetic scenarios. Create test cases representing different situations: a valid request with approvals, a request without approval, a request with unmet conditions, and an attempt to bypass. Verify that the system handles each case

correctly according to the specified policies, and that the audit log records every decision. If the test succeeds, the workflow is confident that the system works as expected, and any case with a real environment reflects real-world complexities that may require human review.

Chapter 41 — Should the Package Be Published on PyPI?

الفصل 41 — هل تُنشر الحزمة على PyPI؟

Deeper Conceptual Explanation

الفكرة المركزية: التصميم، الاختبار، بناء التوزيع، الصدق بشأن حالة النشر.

شرح أعمق للمفهوم

Public release is not merely a button to press. For educational software related to safety, one must consider whether an example could shift from teaching material to a misuse-enabling tool and whether documentation establishes clear boundaries. Restricting distribution can therefore be both an engineering and an ethical decision.

النشر العام ليس مجرد ضغط زر. في البرمجيات التعليمية ذات الصلة بالسلامة، يجب فحص ما إذا كان المثال قد يتحول من مادة تعليمية إلى أداة قابلة لإساءة الاستخدام، وما إذا كانت الوثائق تضع حدوداً واضحة. أحياناً يكون عدم النشر أو تقييد التوزيع قراراً هندسياً وأخلاقياً معاً.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Distinguishing Design from Actual Publication

1. التمييز بين التصميم والنشر الفعلي

One common mistake in research projects is announcing a software package as if it were published, when it is still in the design or local testing phase. The difference between "can be published" and "has actually been published" is fundamental. Publishing on PyPI requires a documented process including:

أحد الأخطاء الشائعة في المشاريع البحثية هو الإعلان عن حزمة برمجية وكأنها منشورة، بينما هي في الواقع لا تزال في مرحلة التصميم أو الاختبار المحلي. الفرق بين "يمكن نشرها" و"نُشرت فعلاً" هو فرق جوهري. النشر على PyPI يتطلب عملية موثقة تشمل:

- استيفاء متطلبات التوزيع: مثل وجود ملف setup.py أو pyproject.toml، ووصف الحزمة، والترخيص، والاعتماديات.

· اجتياز الاختبارات: التأكد من أن جميع الاختبارات تجتاز في بيئات مختلفة.

· Meeting distribution requirements: such as having a setup.py or pyproject.toml file, package description, license, and dependencies.

· بناء التوزيع: إنشاء ملفات التوزيع (sdist و wheel).

· Passing tests: ensuring that all tests pass in different environments.

· الرفع إلى PyPI: استخدام أدوات مثل twine لرفع الحزمة إلى المستودع العام.

· Building the distribution: creating distribution files (sdist and wheel).

· توثيق الإصدار: تسجيل رقم الإصدار، وتاريخ النشر، وتغييرات هذا الإصدار.

· Uploading to PyPI: using tools like twine to upload the package to the public repository.

· Documenting the release: recording the version number, release date, and changes in that version.

مراحل النشر على PyPI:

1. تصميم الحزمة وتطويرها.

2. اختبار محلي.

3. بناء التوزيع.

4. رفع إلى PyPI.

5. توثيق الإصدار.

The mere presence of code in a repository (such as GitHub) does not mean it is published on PyPI. Actual publication requires additional steps and documentation.

مجرد وجود الكود في مستودع (مثل GitHub) لا يعني أنه منشور على PyPI. النشر الفعلي يتطلب خطوات إضافية وتوثيقاً.

2. Publication Claims as Statements Requiring Evidence

2. ادعاءات النشر كتصريحات تحتاج إلى أدلة

In scientific reports and research papers, the claim that "the package is available on PyPI" must be supported by evidence. This evidence includes:

في التقارير العلمية والأوراق البحثية، الادعاء بأن "الحزمة متاحة على PyPI" يجب أن يكون مدعوماً بأدلة. هذه الأدلة تشمل:

· رابط الحزمة على PyPI: رابط مباشر إلى صفحة الحزمة.

· رقم الإصدار: الإصدار المحدد الذي تم نشره.

· The package link on PyPI: a direct link to the package page.

· تاريخ النشر: متى نُشر هذا الإصدار.

· Version number: the specific version that was published.

· التوثيق: دليل المستخدم، واجهة API، وأمثلة.

· Publication date: when this version was published.

· الاختبارات: نتائج الاختبارات التي تجاوزتها الحزمة.

· Documentation: user guide, API reference, and examples.

· Tests: test results that the package passed.

```
# مثال: ادعاء نشر غير مدعوم بأدلة
# خطأ: "الحزمة متاحة على PyPI"
# صواب: "الحزمة biobrick_security متاحة على PyPI بالإصدار 1.0.0،
# نُشرت في 15-01-2025، واجتازت جميع الاختبارات في بيئات Linux و macOS."
```

In the biobrick_security educational package, any publication claim should be supported by clear documentation, specific versions, and verifiability.

في الحزمة التعليمية biobrick_security، يجب أن يكون أي ادعاء بالنشر مدعوماً بتوثيق واضح وإصدارات محددة وقابلية للتحقق.

3. Should the Educational Package Be Published؟

3. هل يجب نشر الحزمة التعليمية؟

This is an open question, depending on several factors:

هذا سؤال مفتوح، يعتمد على عدة عوامل:

مزايا النشر:

· إمكانية الوصول: يمكن للآخرين تثبيت الحزمة واستخدامها بسهولة.

· الاستشهاد: يمكن الاستشهاد بالحزمة في الأبحاث.

Benefits of publishing:

· المراجعة: يمكن للمجتمع مراجعة الكود وتحسينه.

· Accessibility: others can easily install and use the package.

· التأثير: زيادة تأثير العمل البحثي.

فوائد النشر:

· Citation: the package can be cited in research.

· Review: the community can review and improve the code.

· Impact: increasing the impact of the research work.

مخاطر النشر:

· المسؤولية: يجب صيانة الحزمة وإصلاح الأخطاء.

· التوافق: قد تتغير الاعتماديات وتحتاج إلى تحديثات مستمرة.

· الثقة: إذا كانت الحزمة تعليمية فقط، قد يساء استخدامها في سياقات غير مناسبة.

Risks of publishing:

· السلامة: في حالة الحزم الأمنية، النشر قد يزيد من مخاطر سوء الاستخدام.

· Responsibility: the package must be maintained and bugs fixed.

مخاطر النشر:

· Compatibility: dependencies may change and require continuous updates.

· Trust: if the package is only educational, it may be misused in inappropriate contexts.

· Safety: for security-related packages, publishing may increase the risk of misuse.

In the case of biobrick_security, the package is educational and does not contain real data or operational policies. Publishing it may be useful for education, but it must be accompanied by a clear explanation of its limits and educational nature.

في حالة biobrick_security، الحزمة تعليمية ولا تحتوي على بيانات حقيقية أو سياسات تشغيلية. نشرها قد يكون مفيداً للتعليم، لكن يجب أن يكون مصحوباً بتوضيح واضح لحدودها وطبيعتها التعليمية.

4. Requirements for Publishing on PyPI

4. متطلبات النشر على PyPI

To publish the package on PyPI, several technical and organizational requirements must be met:

لنشر الحزمة على PyPI، يجب استيفاء عدة متطلبات تقنية وتنظيمية:

· اسم فريد: يجب أن يكون اسم الحزمة فريداً على PyPI.

· ملف pyproject.toml أو setup.py: يحتوي على معلومات الحزمة (الاسم، الإصدار، الوصف، المؤلف، الترخيص، الاعتماديات).

· اختبارات ناجحة: يجب أن تجتاز الحزمة جميع الاختبارات في بيئات مختلفة.

· توثيق: دليل مستخدم واضح، ووصف للوظائف، وأمثلة.

· Unique name: the package name must be unique on PyPI.

· ترخيص: تحديد ترخيص مفتوح المصدر (مثل MIT، Apache، GPL).

· pyproject.toml or setup.py file: containing package information (name, version, description, author, license, dependencies).

· README: وصف الحزمة، كيفية التثبيت، والاستخدام الأساسي.

· Successful tests: the package must pass all tests in different environments.

· Documentation: clear user guide, function descriptions, and examples.

· License: specifying an open-source license (e.g., MIT, Apache, GPL).

· README: package description, installation instructions, and basic usage.

خطوات النشر الأساسية:

1. إنشاء حساب على PyPI.

2. إعداد ملفات التوزيع.

3. رفع الحزمة باستخدام twine.

4. التحقق من صفحة الحزمة على PyPI.

5. توثيق الإصدار في سجل التغييرات.

5. Honesty About Publication Status

Honesty about publication status means not exaggerating claims. If the package is still in development or testing, this should be clarified. If it is only educational, this should be clearly stated. False or misleading publication claims harm scientific credibility and may lead to legal or ethical consequences.

5. الصدق بشأن حالة النشر

الصدق بشأن حالة النشر يعني عدم المبالغة في الادعاءات. إذا كانت الحزمة لا تزال في مرحلة التطوير أو الاختبار، يجب توضيح ذلك. إذا كانت تعليمية فقط، يجب ذكر ذلك بوضوح. ادعاءات النشر الكاذبة أو المضللة تضر بالمصداقية العلمية وقد تؤدي إلى عواقب قانونية أو أخلاقية.

مبادئ الصدق في النشر:

1. لا تعلن عن النشر قبل النشر الفعلي.

2. وضح حالة الحزمة (تعليمية، تجريبية، إنتاجية).

3. وثق الإصدارات والتغييرات.

4. قدم أدلة على النشر (رابط، إصدار، تاريخ).

5. لا تخلط بين التصميم والنشر.

6. The Security Layer: Stopping When in Doubt

In the context of the educational package, and even in the publication decision, the fail-closed principle applies: if the workflow is not sure about the publication status or the package's suitability for publication, the safe behavior is to stop and request review. The absence of authorization or trusted reference should not become implicit approval. This also applies to the publication decision: there should be explicit approval from relevant parties, documentation of the decision, and reviewability.

6. الطبقة الأمنية: التوقف عند الشك

في سياق الحزمة التعليمية، وحتى في قرار النشر، ينطبق مبدأ fail-closed: إذا لم تكن متأكداً من حالة النشر أو مناسبة الحزمة للنشر، فالسلوك الآمن هو التوقف وطلب المراجعة. لا ينبغي أن يتحول غياب الإذن أو المرجع الموثوق إلى سماح ضمني. هذا ينطبق أيضاً على قرار النشر: يجب أن يكون هناك موافقة صريحة من الجهات المعنية، وتوثيق للقرار، وإمكانية للمراجعة.

```
# مثال: قرار النشر مع تطبيق fail-closed
def decide_publication(package, approval, review_log):
    if not approval:
        review_log.append("فشل: لا توجد موافقة على النشر")
        return {"decision": "denied", "reason": "no explicit approval"}
    if not package.ready_for_publication():
        review_log.append("فشل: الحزمة غير جاهزة للنشر")
        return {"decision": "hold", "reason": "package not ready"}
    review_log.append("نجاح: تمت الموافقة على النشر")
    return {"decision": "approved", "reason": "ready for publication"}
```

الدوال المعرفة هنا (decide_publication) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة.

القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

7. Practical Application: Should biobrick_security Be Published؟

Let us apply this question to the biobrick_security educational package:

7. التطبيق العملي: هل تنشر biobrick_security؟

دعنا نطبق هذا السؤال على حزمة biobrick_security التعليمية:

تقييم biobrick_security للنشر:

- هل الحزمة مكتملة؟ نعم (من الناحية التعليمية).

- هل تجتاز الاختبارات؟ نعم (في بيئات الاختبار).

- هل لديها توثيق كافٍ؟ نعم (دليل تعليمي).

- هل هي جاهزة للإنتاج؟ لا (تعليمية فقط).

- هل هناك طلب عليها؟ قد يكون، للمجتمع التعليمي.

- هل النشر آمن؟ نعم، لأنها لا تحتوي على بيانات حساسة.

القرار: يمكن نشرها كحزمة تعليمية، مع توضيح واضح أنها للتعليم فقط، وليست أداة تشغيلية.

This assessment shows that publication is possible, but must be accompanied by complete transparency about the package's nature and limits.

هذا التقييم يوضح أن النشر ممكن، لكن يجب أن يكون مصحوباً بشفافية تامة حول طبيعة الحزمة وحدودها.

Chapter Summary

خلاصة الفصل

The question "Should the package be published on PyPI?" is not just a technical question, but a test of scientific honesty. A distinction must be made between designing the package, testing it locally, building its distribution, and actually publishing it on PyPI. Publication claims must be supported by evidence: a link, version, date, and documentation. In the case of the biobrick_security educational package, publication is possible, but must be accompanied by a clear explanation of its educational nature and limits. In all cases, the fail-closed principle applies: if

the workflow is unsure, stop and request review. The next chapter will move to Part 11 — RNA-seq: From Sequence to Gene Expression.

السؤال "هل تُنشر الحزمة على PyPI؟" ليس مجرد سؤال تقني، بل هو اختبار للأمانة العلمية. يجب التمييز بين تصميم الحزمة، واختبارها محلياً، وبناء توزيعها، ونشرها فعلاً على PyPI. ادعاءات النشر يجب أن تكون مدعومة بأدلة: رابط، إصدار، تاريخ، وتوثيق. في حالة الحزمة التعليمية biobrick_security، النشر ممكن، لكن يجب أن يكون مصحوباً بتوضيح واضح لطبيعتها التعليمية وحدودها. وفي جميع الأحوال، ينطبق مبدأ fail-closed: إذا كنت غير متأكد، توقف واطلب المراجعة. الفصل القادم سينتقل إلى الباب 11 — RNA-seq: من التسلسل إلى التعبير الجيني.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Should the Package Be Published on PyPI?" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "هل تُنشر الحزمة على PyPI؟" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "publishing the package on PyPI means it is production-ready." This is incorrect; publication means public availability, but it does not mean production readiness. The correct statement is: "publishing the package on PyPI means it is available to others, but its production readiness depends on code quality, documentation, testing, and API stability." A simple test: Practical check: "Has the package undergone security review and extensive testing before publication?".

مثال على التجاوز: القول إن "نشر الحزمة على PyPI يعني أنها جاهزة للإنتاج". هذا غير صحيح؛ النشر يعني التوافر العام، لكنه لا يعني الجاهزية للإنتاج. الصحيح هو: "نشر الحزمة على PyPI يعني أنها متاحة للآخرين، لكن جاهزيتها للإنتاج تعتمد على جودة الكود، والتوثيق، والاختبارات، واستقرار الواجهة البرمجية." اختبار بسيط: فحص تطبيقي: "هل مرت الحزمة بمراجعة أمنية وتجارب مكثفة قبل النشر؟".

Example: A package being on PyPI is a computational representation of availability. Claiming that "this package guarantees biosafety" is a biological claim not proven by PyPI availability. Availability means the package exists, but it does not mean effectiveness or safety. Practical test: before relying on a package for biological purposes, check its documentation, tests, and reviews.

مثال: وجود حزمة على PyPI هو تمثيل حسابي للتوافر. الادعاء بأن "هذه الحزمة تضمن السلامة البيولوجية" هو ادعاء بيولوجي لا يثبته التوافر على PyPI. التوافر يعني أن الحزمة موجودة، لكنه لا يعني فعاليتها أو سلامتها. اختبار عملي: قبل أن تعتمد على حزمة لأغراض بيولوجية، تحقق من توثيقها، واختباراتها، ومراجعاتها.

مثال تطبيقي على قرار نشر حزمة:

· المدخلات: الكود، الاختبارات، التوثيق، الترخيص، سياسات النشر.

· الافتراضات: أن الكود يعمل كما هو متوقع، وأن الاختبارات تغطي الحالات الأساسية، وأن التوثيق كافٍ للمستخدمين.

· المخرجات: حزمة متاحة على PyPI، مع إصدار موثق.

· القيود: النشر لا يضمن الجودة، ولا يضمن الملاءمة لجميع الاستخدامات، ولا يضمن الصيانة المستمرة.

Applied example of a package publication decision:

اختبار: قبل نشر حزمة، تأكد من استيفاء جميع المتطلبات التقنية والقانونية، وأن يوجد في المثال خطة للصيانة.

· Inputs: code, tests, documentation, license, publication policies.

· Assumptions: that the code works as expected, that tests cover basic cases, and that documentation is sufficient for users.

· Outputs: a package available on PyPI, with a documented release.

· Limitations: publication does not guarantee quality, nor suitability for all uses, nor ongoing maintenance.

Test: before publishing a package, ensure all technical and legal requirements are met, and that the workflow has a maintenance plan.

عند نشر حزمة، يجب تسجيل:

· المصدر: الكود المصدري، تاريخ التطوير، الإصدارات السابقة.

· الإصدار: رقم الإصدار المنشور، وتاريخ النشر.

· المعلمات: إصدارات الاعتماديات، إصدار Python المستخدم.

· التحويلات: أي تعديلات على الكود قبل النشر (مثل تنظيف، تحسين، إزالة بيانات حساسة).

When publishing a package, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لمستخدم بعد عام، هل يستطيع فهم ما نُشر، ومتى، وما هي الاعتماديات المطلوبة؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the source code, development history, previous versions.

· Version: the published version number, and release date.

· Parameters: dependency versions, Python version used.

· Transformations: any modifications to the code before publication (e.g., cleaning, optimization, removing sensitive data).

Test: Practical check: "If I gave this record to a user a year later, could they understand what was published, when, and what dependencies are required?" If the answer is no, the documentation is insufficient.

قبل نشر أي حزمة، اختبر عملية النشر على بيئة اختبار (مثل TestPyPI). هذا يسمح بالتحقق من أن عملية الرفع تعمل، وأن الملفات صحيحة، وأن الوصف يظهر بشكل صحيح، دون التأثير على المستودع العام. إذا نجح الاختبار على TestPyPI، فأنت واثق من أن النشر الفعلي سيعمل كما هو متوقع.

Before publishing any package, test the publication process on a testing environment (such as TestPyPI). This allows verification that the upload process works, that files are correct, and that the description appears correctly, without affecting the public repository. If the test succeeds on TestPyPI, the workflow is confident that the actual publication will work as expected.

Part 11 — RNA-seq: From Sequence to Gene Expression

الباب 11 — RNA-seq: من التسلسل إلى التعبير الجيني

الفكرة المركزية: سلسلة القياس، المكتبة، القراءات، التقدير، النمذجة.

Chapter 42 — What Does RNA-seq Measure?

الفصل 42 — ما الذي يقيسه RNA-seq؟

شرح أعمق للمفهوم

Deeper Conceptual Explanation

RNA-seq يقيس إشارات مرتبطة بالـRNA الموجود في العينة بعد خطوات إعداد وتجهيز وقراءة وحساب. ما ينتج في النهاية يعتمد على التصميم التجريبي، وجودة القراءة، وطريقة العد والتطبيع. لذلك يجب ألا تُقرأ قائمة الجينات المختلفة وكأنها قائمة مباشرة بالوظائف السببية.

RNA-seq measures signals related to RNA present in a sample after preparation, sequencing, and computational processing. The final quantities depend on experimental design, read quality, counting, and normalization. A list of differentially expressed genes should therefore not be read as a direct list of causal functions.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. ما هو RNA-seq؟

1. What Is RNA-seq؟

RNA-seq (RNA sequencing) is a technique used to determine the quantity and type of RNA in a biological sample. Instead of measuring proteins directly, RNA-seq measures RNA molecules transcribed from DNA, giving a snapshot of which genes were active at the time of sample collection. RNA-seq is not a single measurement, but a chain of interconnected steps, and each step affects the final result.

RNA-seq (RNA sequencing) هي تقنية تستخدم لتحديد كمية ونوع الـ RNA في عينة بيولوجية. بدلاً من قياس البروتينات مباشرة، يقيس RNA-seq جزيئات الـ RNA التي تُنسخ من DNA، مما يعطي صورة عن الجينات التي كانت نشطة في وقت جمع العينة. RNA-seq ليس مقياساً واحداً، بل هو سلسلة من الخطوات المترابطة، وكل خطوة تؤثر على النتيجة النهائية.

ما يقيسه RNA-seq:

- وفرة جزيئات الـ RNA في العينة.

- ليس مباشرة التعبير البروتيني (قد يكون هناك تأخير أو تنظيم).

- يعكس حالة الخلية في لحظة معينة.

- يعتمد على سلسلة من القرارات التجريبية والحاسوبية.

2. The Measurement Chain: From Sample to Data

RNA-seq is a chain of steps, which can be divided into main stages:

2. سلسلة القياس: من العينة إلى البيانات

RNA-seq هو سلسلة من الخطوات، يمكن تقسيمها إلى مراحل رئيسية:

المرحلة الأولى: العينة البيولوجية وإعداد المكتبة.

تبدأ العملية باستخلاص الـ RNA من العينة. يُحول الـ RNA إلى DNA مكمل (cDNA) باستخدام إنزيم النسخ العكسي. ثم تُضاف محولات (adapters) لتسمح بتسلسل الجزيئات. هذا الإعداد يؤثر على تمثيلية العينة وجودة البيانات.

المرحلة الثانية: التسلسل (Sequencing).

يُمرر cDNA عبر جهاز تسلسل (مثل Illumina) لتوليد ملايين القراءات القصيرة. طول القراءات يختلف حسب التقنية (عادة 50-300 قاعدة). جودة القراءات تُقاس بدرجات الجودة (Phred scores).

المرحلة الثالثة: مراقبة الجودة (Quality Control).

تُفحص القراءات الخام للتأكد من جودتها. تُزال القراءات منخفضة الجودة، وتُقص المحولات، وتُعالج التحيزات التقنية.

المرحلة الرابعة: المحاذاة أو التجميع (Alignment or Assembly).

تُحاذى القراءات على جينوم مرجعي (إذا كان متوفراً) أو تُجمّع لتكوين تسلسلات جديدة. هذه الخطوة تحدد أي جزء من الجينوم أنتج كل قراءة.

المرحلة الخامسة: تقدير الكميات (Quantification).

Stage One: Biological Sample and Library Preparation.

تُحسب أعداد القراءات التي تتبع كل جين. هذه الأعداد الخام تحتاج إلى تطبيع لتصحيح اختلافات عمق التسلسل وأطوال الجينات.

The process begins with RNA extraction from the sample. RNA is converted to complementary DNA (cDNA) using reverse transcriptase. Adapters are then added to allow sequencing of the molecules. This preparation affects sample representation and data quality.

المرحلة السادسة: النمذجة الإحصائية (Statistical Modeling).

Stage Two: Sequencing

تُستخدم نماذج إحصائية لاختبار الفروق في التعبير بين الظروف (مثل المريض مقابل السليم)، مع الأخذ في الاعتبار التباين التقني والبيولوجي.

The cDNA is run through a sequencer (e.g., Illumina) to generate millions of short reads. Read length varies by technology (usually 50-300 bases). Read quality is measured by quality scores (Phred scores).

Stage Three: Quality Control

Raw reads are checked for quality. Low-quality reads are removed, adapters are trimmed, and technical biases are addressed.

Stage Four: Alignment or Assembly.

Reads are aligned to a reference genome (if available) or assembled to form new sequences. This step determines which part of the genome produced each read.

Stage Five: Quantification

The number of reads mapping to each gene is counted. These raw counts need normalization to correct for differences in sequencing depth and gene length.

Stage Six: Statistical Modeling.

Statistical models are used to test for expression differences between conditions (e.g., patient vs. healthy), accounting for technical and biological variation.

3. Sources of Variation in RNA-seq

3. مصادر التباين في RNA-seq

Variation in RNA-seq results can arise from several sources, including biological, technical, and processing sources:

التباين في نتائج RNA-seq يمكن أن ينشأ من عدة مصادر، منها البيولوجية والتقنية والمعالجة:

التباين البيولوجي (Biological variation) : اختلافات حقيقية بين العينات بسبب نوع الخلية، الحالة المرضية، الاستجابة للعلاج، أو التباين الطبيعي بين الأفراد.

Biological variation: real differences between samples due to cell type, disease state, treatment response, or natural variation between individuals.

التباين التقني (Technical variation) : ينشأ من خطوات التجربة: جودة استخلاص الـ RNA، كفاءة إعداد المكتبة، اختلافات بين دفعات التسلسل، أداء جهاز التسلسل.

Technical variation: arises from experimental steps: RNA extraction quality, library preparation efficiency, differences between sequencing batches, and sequencer performance.

تباين المعالجة (Processing variation) : ينشأ من اختيارات المحاذاة، التطبيع، طريقة تقدير الكميات، النموذج الإحصائي المستخدم.

Processing variation: arises from choices in alignment, normalization, quantification method, and statistical model.

التباين الكلي = تباين بيولوجي + تباين تقني + تباين معالجة

Understanding these sources of variation is the foundation for designing a reliable RNA-seq experiment and interpreting its results.

فهم مصادر التباين هذه هو أساس تصميم تجربة RNA-seq موثوقة وتفسير نتائجها.

4. Connecting Measurement to Statistical Modeling

4. ربط القياس بالنمذجة الإحصائية

Raw data from RNA-seq are read counts. But these counts do not directly reflect expression levels. They must be normalized to correct for:

البيانات الخام من RNA-seq هي أعداد قراءات. لكن هذه الأعداد لا تعكس مباشرة مستويات التعبير. يجب تطبيعها لتصحيح:

· عمق التسلسل (Sequencing depth) : العينات ذات التسلسل الأعمق تنتج أعداداً أكبر، حتى لو كان التعبير نفسه.

· Sequencing depth: samples with deeper sequencing produce higher counts, even if expression is the same.

· طول الجين (Gene length) : الجينات الأطول تنتج عدداً أكبر من القراءات، حتى لو كان التعبير نفسه.

· Gene length: longer genes produce more reads, even if expression is the same.

· تأثيرات الدفعات (Batch effects) : اختلافات بين دفعات التسلسل قد تؤثر على الأعداد.

· Batch effects: differences between sequencing batches may affect counts.

Normalization methods such as TPM (Transcripts Per Million), RPKM (Reads Per Kilobase per Million), and DESeq2's median-of-ratios attempt to correct for these factors. The choice of normalization method affects results and should be based on the data type and biological question.

طرق التطبيع مثل TPM (Transcripts Per Million) و RPKM (Reads Per Kilobase per Million) و DESeq2's median-of-ratios تحاول تصحيح هذه العوامل. اختيار طريقة التطبيع يؤثر على النتائج، ويجب أن يكون مستنداً إلى نوع البيانات والسؤال البيولوجي.

```
# مثال تعليمي: تأثير عمق التسلسل على الأعداد
sample1_counts = [100, 200, 300] # عمق تسلسل منخفض
sample2_counts = [200, 400, 600] # عمق تسلسل مرتفع (ضعف الأول)
# بدون تطبيع، تبدو العينة الثانية وكأن تعبيرها أعلى
# مع التطبيع، تصبح النسب متساوية
```

5. Limits of Interpreting RNA-seq Results

RNA-seq results are estimates, not absolute facts. They must be interpreted in context:

5. حدود تفسير نتائج RNA-seq

نتائج RNA-seq هي تقديرات، وليست حقائق مطلقة. يجب تفسيرها في سياقها:

· الارتباط لا يساوي السببية: تغير التعبير لا يعني أن الجين يسبب المرض.

· Correlation does not equal causation: changes in expression do not mean the gene causes the disease.

· مستوى الـ RNA لا يساوي مستوى البروتين: قد لا يترجم الـ RNA إلى بروتين، أو قد يُعدل البروتين بعد الترجمة.

· RNA level does not equal protein level: RNA may not be translated into protein, or protein may be post-translationally modified.

· التعبير يختلف بين الخلايا: العينة تتكون من خلايا متعددة، ومتوسط التعبير قد يخفي اختلافات بين أنواع الخلايا.

· Expression varies between cells: the sample consists of multiple cells, and average expression may hide differences between cell types.

· الحساسية والنوعية: قد تفوت بعض الجينات منخفضة التعبير (سلبيات كاذبة) أو تكتشف إشارات غير حقيقية (إيجابيات كاذبة).

· Sensitivity and specificity: some low-expression genes may be missed (false negatives) or false signals may be detected (false positives).

مبدأ أساسي: نتائج RNA-seq هي فرضيات، وليست حقائق نهائية. تحتاج إلى تحقق تجريبي مستقل.

6. Practical Application: Simulating RNA-seq Data

In the educational context, RNA-seq data can be simulated using synthetic data to understand principles. The goal is not to analyze real data, but to understand the steps.

6. التطبيق العملي: محاكاة بيانات RNA-seq

في السياق التعليمي، يمكن محاكاة بيانات RNA-seq باستخدام بيانات اصطناعية لفهم المبادئ. الهدف ليس تحليل بيانات حقيقية، بل فهم الخطوات.

```
# محاكاة تعليمية لبيانات RNA-seq
import numpy as np
def simulate_rnaseq_data(n_genes=100, n_samples=10, n_conditions=2):
    """
```

محاكاة بيانات RNA-seq اصطناعية للتعليم.

```
    """
    # إنشاء أعداد قراءات عشوائية من توزيع بواسون (محاكاة)
    np.random.seed(42)
    counts = np.random.poisson(lam=50, size=(n_genes, n_samples))
# إضافة تأثير دفعة (محاكاة)
    batch_effect = np.random.normal(0, 5, size=n_samples)
    for j in range(n_samples):
        counts[:, j] += batch_effect[j]
# إضافة اختلاف بيولوجي اصطناعي (بعض الجينات تختلف بين الحالتين)
    condition1 = np.random.choice([0, 1], size=n_genes, p=[0.7, 0.3])
    for i in range(n_genes):
        if condition1[i] == 1:
            counts[i, :5] *= 2  # زيادة في الحالة الأولى

    return {
```

```
    "counts": counts,
    "normalized": normalized,
    "condition": np.array([0]*5 + [1]*5),
    "n_genes": n_genes,
    "n_samples": n_samples
}
```

```
# تطبيع بسيط (محاكاة)
normalized = counts / counts.sum(axis=0, keepdims=True) * 1e6  # CPM محاكاة
# تشغيل المحاكاة
data = simulate_rnaseq_data()
print(f"شكل بيانات الأعداد: {data['counts'].shape}")
print(f"شكل البيانات المطبعة: {data['normalized'].shape}")
print(f"الحالات: {data['condition']}")
```

This educational simulation illustrates the principle: RNA-seq data consists of read counts, needs normalization, and reflects biological and technical effects.

7. Reproducibility in RNA-seq Analysis

هذه المحاكاة التعليمية توضح المبدأ: بيانات RNA-seq تتكون من أعداد قراءات، وتحتاج إلى تطبيع، وتعكس تأثيرات بيولوجية وتقنية.

RNA-seq analysis must be reproducible. This requires:

7. إعادة الإنتاج في تحليل RNA-seq

تحليل RNA-seq يجب أن يكون قابلاً لإعادة الإنتاج. هذا يتطلب:

· توثيق جميع الخطوات: من استخلاص الـ RNA إلى التحليل الإحصائي.

· Documenting all steps: from RNA extraction to statistical analysis.

· تسجيل الإصدارات: إصدارات الأدوات البرمجية، وقواعد البيانات المرجعية (الجينوم، التعليقات التوضيحية).

· Recording versions: software tool versions, reference databases (genome, annotations).

· تسجيل المعلمات: معلمات المحاذاة، التطبيع، والنمذجة الإحصائية.

· Recording parameters: alignment, normalization, and statistical modeling parameters.

· مشاركة البيانات والكود: إتاحة البيانات الخام والكود المستخدم.

· Sharing data and code: making raw data and code available.

إعادة الإنتاج = بيانات خام + كود + توثيق + إصدارات + معلمات

If a biological source, library version, or normalization method changes, the resulting difference is not an error to be hidden; it is information to be interpreted. Hence, provenance records become part of the scientific method.

إذا تغير مصدر حي أو إصدار مكتبة أو طريقة التطبيع، فاختلاف الناتج ليس خطأ يجب إخفاؤه؛ بل معلومة يجب تفسيرها. ومن هنا تصبح سجلات provenance جزءاً من المنهج العلمي.

Chapter Summary

خلاصة الفصل

RNA-seq is a chain of experimental and computational decisions, not just a single step. It begins with biological material, passes through library preparation, sequencing, quality control, alignment or assembly, quantification, and statistical modeling. Each layer introduces its assumptions and sources of variation. The final result is the product of this entire chain. Understanding this chain is the foundation of reliable RNA-seq analysis. Variation arises from biological, technical, and processing sources. Normalization and statistical modeling are tools to address this variation, but they do not eliminate the need for good experimental design and careful documentation. RNA-seq results are hypotheses, not final facts, and require independent verification. The next chapter will move to read quality in RNA-seq.

RNA-seq هو سلسلة من القرارات التجريبية والحاسوبية، وليس مجرد خطوة واحدة. تبدأ بالمادة البيولوجية، وتمر بإعداد المكتبة، والتسلسل، ومراقبة الجودة، والمحاذاة أو التجميع، وتقدير الكميات، والنمذجة الإحصائية. كل طبقة تضيف افتراضاتها ومصادر التباين. النتيجة النهائية هي نتاج هذه السلسلة بأكملها. فهم هذه السلسلة هو أساس تحليل RNA-seq موثوق. التباين ينشأ من مصادر بيولوجية وتقنية ومعالجة. التطبيع والنمذجة الإحصائية أدوات للتعامل مع هذا التباين، لكنها لا تلغي الحاجة إلى التصميم التجريبي الجيد والتوثيق الدقيق. نتائج RNA-seq هي فرضيات، وليست حقائق نهائية، وتحتاج إلى تحقق مستقل. الفصل القادم سينتقل إلى جودة القراءات في RNA-seq.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "What Does RNA-seq Measure?" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "ما الذي يقيسه RNA-seq؟" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "RNA-seq measures protein expression directly." This is incorrect; RNA-seq measures RNA abundance, not protein. The correct statement is: "RNA-seq measures the quantities of RNA molecules in the sample, which reflect gene activity, but do not directly reflect protein levels or

function." A simple test: Practical check: "Does this analysis tell me about the actual protein in the cell, or about the RNA that may be translated to protein?".

مثال على التجاوز: القول إن "RNA-seq يقيس التعبير البروتيني مباشرة". هذا غير صحيح؛ RNA-seq يقيس وفرة الـ RNA، وليس البروتين. الصحيح هو: "RNA-seq يقيس كميات جزيئات الـ RNA في العينة، والتي تعكس نشاط الجينات، لكنها لا تعكس مباشرة مستويات البروتين أو وظيفته." اختبار بسيط: فحص تطبيقي: "هل هذا التحليل يخبرني عن البروتين الفعلي في الخلية، أم عن الـ RNA الذي قد يترجم إلى بروتين؟"ـ

Example: RNA-seq analysis results giving a list of differentially expressed genes with p-values is a computational representation. Claiming that "these genes cause the disease" is a biological claim not proven by statistical analysis alone. The analysis identifies genes with changed expression, but proving a causal role requires independent experiments. Practical test: before inferring a biological role from RNA-seq results, Practical check: "Is there independent evidence (such as gene knockout experiments, or other studies) supporting this role?".

مثال: نتيجة تحليل RNA-seq تعطي قائمة بالجينات ذات التعبير التفاضلي مع قيم p هي تمثيل حسابي. الادعاء بأن "هذه الجينات تسبب المرض" هو ادعاء بيولوجي لا يثبته التحليل الإحصائي وحده. التحليل يحدد جينات ذات تغير في التعبير، لكن إثبات الدور السببي يحتاج إلى تجارب مستقلة. اختبار عملي: قبل أن تستنتج دوراً بيولوجياً من نتائج RNA-seq، فحص تطبيقي: "هل هناك أدلة مستقلة (مثل تجارب تعطيل الجين، أو دراسات أخرى) تدعم هذا الدور؟"ـ

مثال تطبيقي على تحليل RNA-seq:

· المدخلات: ملفات FASTQ (قراءات خام)، جينوم مرجعي، تعليقات توضيحية، معلمات المحاذاة والتطبيعـ

· الافتراضات: أن القراءات تمثل العينة بدقة، وأن الجينوم المرجعي مناسب، وأن طريقة التطبيع تعكس الواقع البيولوجيـ

· المخرجات: أعداد قراءات لكل جين، قيم تطبيع، قيم p، قيم p معدلة، وحجم التأثيرـ

· القيود: التحليل يعتمد على جودة القراءات، وكفاءة المحاذاة، وملاءمة النموذج الإحصائي. قد تفوت الجينات منخفضة التعبير، وقد تكون هناك إيجابيات كاذبةـ

Applied example of RNA-seq analysis:

اختبار: قبل تحليل RNA-seq، تحقق من جودة القراءات (FastQC)، وتأكد من أن الجينوم المرجعي مناسب، واستخدم رسوم تشخيصية (مثل PCA) للكشف عن تأثيرات الدفعاتـ

· Inputs: FASTQ files (raw reads), reference genome, annotations, alignment and normalization parameters.

· Assumptions: that reads accurately represent the sample, that the reference genome is appropriate, and that the normalization method reflects biological reality.

· Outputs: read counts per gene, normalized values, p-values, adjusted p-values, and effect sizes.

· Limitations: analysis depends on read quality, alignment efficiency, and statistical model appropriateness. Low-expression genes may be missed, and there may be false positives.

Test: before RNA-seq analysis, check read quality (FastQC), ensure the reference genome is appropriate, and use diagnostic plots (e.g., PCA) to detect batch effects.

عند تحليل RNA-seq، يجب تسجيل ما يلي:

· المصدر: البيانات الخام (ملفات FASTQ، معرفات العينات)، الجينوم المرجعي (الإصدار)، التعليقات التوضيحية (الإصدار).

· الإصدار: إصدار أدوات المحاذاة (مثل STAR، HISAT2)، التطبيع (مثل DESeq2، edgeR)، وإصدار R/Bioconductor.

· المعلمات: معلمات المحاذاة، طريقة التطبيع، النموذج الإحصائي، العتبات المستخدمة.

When analyzing RNA-seq, record:

· Provenance: raw data (FASTQ files, sample identifiers), reference genome (version), annotations (version).

· Version: alignment tool versions (e.g., STAR, HISAT2), normalization (e.g., DESeq2, edgeR), and R/Bioconductor version.

· Parameters: alignment parameters, normalization method, statistical model, and thresholds used.

قبل تحليل بيانات RNA-seq حقيقية، اختبر مسار العمل على بيانات اصطناعية. استخدم بيانات محاكاة ذات خصائص معروفة (عدد الجينات، عدد العينات، اختلافات بيولوجية معروفة). طبّق نفس خطوات التحليل (المحاذاة، التطبيع، النمذجة) وتأكد من أن النتائج تعكس الاختلافات المعروفة. إذا نجح الاختبار، فأنت واثق من أن مسار العمل يعمل كما هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس خصائص البيانات وليس خطأ في الإعدادات.

Before analyzing real RNA-seq data, test the workflow on synthetic data. Use simulated data with known properties (number of genes, number of samples,

known biological differences). Apply the same analysis steps (alignment, normalization, modeling) and verify that the results reflect the known differences. If the test succeeds, the workflow is confident that the workflow works as expected, and any result with real data reflects the data's properties, not an error in the settings.

Chapter 43 — Read Quality

الفصل 43 — جودة القراءات

الفكرة المركزية: QC، bias، trimming، quality scores، قرارات المعالجة.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

الجودة في RNA-seq ليست رقماً واحداً. تشمل جودة القواعد، والقراءات المكررة، والتلوث، وتوزيع الأطوال، ومشكلات المحاذاة أو التخصيص. فحص الجودة يحدد ما إذا كانت البيانات مناسبة للخطوة التالية، ولا يضمن وحده صحة الاستنتاج النهائي.

Quality in RNA-seq is not a single number. It includes base quality, duplication, contamination, read-length distributions, and alignment or assignment problems. Quality control determines whether data are suitable for the next stage; it does not by itself guarantee the final inference.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Why Is Read Quality Important؟

1. لماذا جودة القراءات مهمة؟

Sequencing reads are not perfect. Sequencers produce errors, and the quality of some bases may be low due to technical or chemical factors. These errors can affect:

قراءات التسلسل ليست مثالية. أجهزة التسلسل تنتج أخطاء، وقد تكون جودة بعض القواعد منخفضة بسبب عوامل تقنية أو كيميائية. هذه الأخطاء يمكن أن تؤثر على:

· المحاذاة (Alignment): القراءات منخفضة الجودة قد لا تحاذي بشكل صحيح على الجينوم المرجعي، أو قد تحاذي في مواقع خاطئة.

· Alignment: low-quality reads may not align correctly to the reference genome, or may align to incorrect positions.

· تقدير الكميات (Quantification): الأخطاء في القراءات قد تؤدي إلى عدّ قراءات غير صحيحة لجينات معينة.

· Quantification: errors in reads may lead to incorrect counting of reads for specific genes.

· الاستنتاجات البيولوجية (Biological conclusions): الأخطاء قد تؤدي إلى اكتشاف جينات مختلفة بشكل خاطئ (إيجابيات كاذبة) أو تفويت جينات حقيقية (سلبيات كاذبة).

· Biological conclusions: errors may lead to false discovery of differentially expressed genes (false positives) or missing real genes (false negatives).

مبدأ أساسي: جودة المدخلات تحدد جودة المخرجات.

"Garbage in, garbage out."

Therefore, checking read quality and processing it appropriately is a mandatory step in any reliable RNA-seq analysis.

لذلك، فحص جودة القراءات ومعالجتها بشكل مناسب هو خطوة إلزامية في أي تحليل RNA-seq موثوق.

2. Read Quality Indicators

Several indicators are used to assess the quality of raw reads. The most important are:

2. مؤشرات جودة القراءات

درجات الجودة (Quality Scores — Phred Scores).

The quality score (Phred score) is a measure of the probability that a base read by the sequencer is incorrect. It is calculated as:

هناك عدة مؤشرات تستخدم لتقييم جودة القراءات الخام. أهمها:

درجة الجودة (Phred score) هي مقياس احتمالية أن تكون القاعدة التي قرأها جهاز التسلسل خاطئة. تُحسب كالتالي:

```
Q = -10 * log10(P)
```

Where P is the error probability. A Q20 score means a 1% error probability, and Q30 means 0.1% error probability. Typically, bases with a score below 20 (or 30 in some settings) are considered low quality.

حيث P هي احتمالية الخطأ. درجة Q20 تعني احتمال خطأ 1%، و Q30 تعني احتمال خطأ 0.1%. عادةً، تُعتبر القواعد ذات درجة أقل من 20 (أو 30 في بعض الإعدادات) منخفضة الجودة.

مثال:

قاعدة بدرجة Q10: احتمال خطأ 10%

قاعدة بدرجة Q20: احتمال خطأ 1%

قاعدة بدرجة Q30: احتمال خطأ 0.1%

قاعدة بدرجة Q40: احتمال خطأ 0.01%

توزيع الجودة عبر القراءة (Quality Distribution Across the Read).

Often, base quality decreases along the read. At the beginning of the read, quality may be high, then gradually decline towards the end. This is due to technical factors in the sequencing process.

نسبة GC (GC Content)

GC content is the percentage of G and C bases in the read or sample. Deviation from the expected GC content may indicate contamination or library preparation bias.

غالباً ما تنخفض جودة القواعد مع تقدم القراءة. في بداية القراءة، قد تكون الجودة عالية، ثم تتراجع تدريجياً نحو النهاية.

هذا بسبب عوامل تقنية في عملية التسلسل.

نسبة GC (Guanine-Cytosine content) هي النسبة المئوية للقواعد G و C في القراءة أو في العينة. انحراف

نسبة GC عن المتوقع قد يشير إلى تلوث أو تحيز في إعداد المكتبة.

High levels of identical reads may indicate over-amplification during library preparation (PCR duplicates), affecting quantification.

تكرارات الكائنات المتسلسلة (Sequence Duplication Levels).

وجود كميات كبيرة من القراءات المتطابقة قد يشير إلى تضخيم زائد أثناء إعداد المكتبة (PCR duplicates)، مما

يؤثر على تقدير الكميات.

المحولات (Adapters)

The presence of adapter sequences at the ends of reads indicates that adapters were not fully trimmed during data processing.

وجود متواليات المحولات (adapters) في نهايات القراءات يشير إلى أن المحولات لم تُقص بشكل كامل أثناء

معالجة البيانات.

3. Recognizing Contamination and Technical Artifacts

Contamination is the presence of genetic material from an unintended source in the sample. This may be from an external source (such as contamination from another sample, or from the organism itself if not fully removed). Technical artifacts include:

3. التعرف على التلوث والمظاهر التقنية

التلوث هو وجود مادة جينية من مصدر غير مقصود في العينة. قد يكون هذا من مصدر خارجي (مثل تلوث من عينة أخرى، أو من الكائن الحي نفسه إذا لم يُزال بشكل كامل). المظاهر التقنية تشمل:

· التحيز في تضخيم PCR (PCR bias): بعض التسلسلات تُضخم أكثر من غيرها أثناء إعداد المكتبة.

· PCR amplification bias: some sequences are amplified more than others during library preparation.

· تحيز المحتوى GC (GC bias): بعض المناطق الغنية بـ GC أو الفقيرة بها قد تُقرأ بكفاءة أقل.

· GC content bias: regions rich or poor in GC may be read with lower efficiency.

· تأثيرات الدفعات (Batch effects): اختلافات بين دفعات التسلسل قد تؤثر على جودة القراءات وكمياتها.

· Batch effects: differences between sequencing batches may affect read quality and quantities.

كيف نكتشف التلوث؟

- مقارنة نسبة GC بالمتوقع للكائن الحي.

- البحث عن تسلسلات من كائنات أخرى (مثل البكتيريا أو الفيروسات).

- فحص توزيع الجودة والكميات بين العينات.

Tools like FastQC and MultiQC provide comprehensive reports on read quality and reveal these issues.

أدوات مثل FastQC و MultiQC توفر تقارير شاملة عن جودة القراءات وتكشف عن هذه المشكلات.

4. Preprocessing Decisions: Trimming and Cleaning

Based on quality reports, we make decisions about processing raw reads. Common decisions include:

4. قرارات المعالجة المسبقة: القص والتنظيف

Trimming: removing parts of the read, usually from the end where quality is low. Adapters can also be trimmed.

بناءً على تقارير الجودة، نتخذ قرارات بشأن معالجة القراءات الخام. القرارات الشائعة تشمل:

Filtering: removing entire reads if their quality is too low, or if they are too short after trimming.

القص (Trimming) : إزالة أجزاء من القراءة، عادةً من النهاية حيث تكون الجودة منخفضة. يمكن أيضاً قص المحولات.

Deduplication: removing identical reads (PCR duplicates) in some types of analysis.

التصفية (Filtering) : إزالة القراءات بأكملها إذا كانت جودتها منخفضة جداً، أو إذا كانت قصيرة جداً بعد القص.

إزالة التكرارات (Deduplication) : إزالة القراءات المتطابقة (PCR duplicates) في بعض أنواع التحليل.

```
# مثال تعليمي: قرارات القص (محاكاة)
def trim_read(read, quality_scores, min_quality=20):
    """
```

قص القراءة بناءً على جودة القواعد (محاكاة تعليمية).

```
    """
    # العثور على أول قاعدة بجودة منخفضة من النهاية
    trim_pos = len(read)
    for i in range(len(quality_scores)-1, -1, -1):
        if quality_scores[i] < min_quality:
            trim_pos = i
```

break

```
    return read[:trim_pos]
```

الدوال المعرفة هنا (trim_read) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

But trimming or cleaning does not always mean improvement. It may lead to:

لكن القص أو التنظيف لا يعني دائماً تحسيناً. قد يؤدي إلى:

· Information loss: biologically useful regions may be trimmed.

· فقد معلومات: قد تُقص مناطق مفيدة بيولوجياً.

· Bias: reads may be trimmed inconsistently between samples, creating artificial differences.

· تحيز: قد يُقص قراءات بطريقة غير متساوية بين العينات، مما يخلق اختلافات مصطنعة.

· Data distribution change: may affect subsequent statistical analysis.

· تغيير توزيع البيانات: قد يؤثر على التحليل الإحصائي اللاحق.

مبدأ: يجب تبرير كل قرار معالجة مسبقة. لماذا اخترت عتبة معينة؟ لماذا قصصت من النهاية وليس من البداية؟ لماذا استبعدت هذه القراءات؟

5. Justifying Preprocessing Decisions

Because preprocessing decisions affect results, they must be documented and justified clearly. This includes:

5. تبرير قرارات المعالجة المسبقة

لأن قرارات المعالجة المسبقة تؤثر على النتائج، يجب توثيقها وتبريرها بوضوح. هذا يشمل:

· تسجيل العتبات المستخدمة: لماذا اخترت Q20 وليس Q30؟

· Recording the thresholds used: why did you choose Q20 and not Q30?

· تسجيل الأدوات والإصدارات: أي إصدار من FastQC أو Trimmomatic أو cutadapt استخدمت؟

· Recording tools and versions: which version of FastQC, Trimmomatic, or cutadapt did you use?

· تسجيل التسلسل الزمني: ما هي الخطوات التي طبقت، وبأي ترتيب؟

· Recording the sequence: what steps were applied, and in what order?

· تسجيل تأثير القرارات: كيف تغيرت أعداد القراءات والجودة بعد المعالجة؟

· Recording the impact of decisions: how did read counts and quality change after processing?

```
# مثال: توثيق قرارات المعالجة
processing_log = {
    "tool": "Trimmomatic",
    "version": "0.39",
    "parameters": {
        "leading": 3,
        "trailing": 3,
        "sliding_window": "4:15",
        "minlen": 36
    },
    "reads_before": 1000000,
    "reads_after": 950000,
    "date": "2025-01-15"
```

```
}
```

This documentation is what allows others to understand what you did, why, and its impact. It is the foundation of reproducibility.

6. Practical Application: Quality Checking Using FastQC (Simulation)

هذا التوثيق هو ما يسمح للآخرين بفهم ما فعلته، ولماذا، وما تأثيره. وهو أساس إعادة الإنتاج.

In practice, we use tools like FastQC to check read quality. Here is an educational simulation of how FastQC works:

6. التطبيق العملي: فحص الجودة باستخدام FastQC (محاكاة)

في الممارسة العملية، نستخدم أدوات مثل FastQC لفحص جودة القراءات. هنا محاكاة تعليمية لكيفية عمل FastQC:

```
# محاكاة تعليمية لفحص الجودة
class FastQCSimulation:
    def __init__(self, reads, quality_scores):
        self.reads = reads
        self.quality_scores = quality_scores
        self.report = {}
def run(self):
        """محاكاة تقرير FastQC."""
        # 1. متوسط الجودة لكل موضع
        avg_quality = []
        for pos in range(len(self.quality_scores[0])):
            pos_qualities = [q[pos] for q in self.quality_scores if pos < len(q)]
            avg_quality.append(sum(pos_qualities) / len(pos_qualities) if pos_qualities else 0)
        self.report["avg_quality"] = avg_quality
# 2. نسبة GC
        gc_contents = []
        for read in self.reads:
            gc = (read.count('G') + read.count('C')) / len(read) * 100
            gc_contents.append(gc)
        self.report["gc_content"] = sum(gc_contents) / len(gc_contents)
# 3. تحديد المناطق منخفضة الجودة
        low_quality_regions = []
        for pos, q in enumerate(avg_quality):
            if q < 20:
                low_quality_regions.append(pos)
```

self.report["low_quality_positions"] = low_quality_regions

4. توصيات

if len(low_quality_regions) > len(avg_quality) * 0.3:

self.report["recommendation"] = "trim_ends"

else:

self.report["recommendation"] = "no_action"

return self.report

```
# محاكاة بيانات
reads = ["ATGCGTACGTAGCTAGCTAGCTAGCTAGC" for _ in range(100)]
quality = [[30]*10 + [20]*10 + [10]*10 for _ in range(100)]  # جودة تتناقص
qc = FastQCSimulation(reads, quality)
report = qc.run()
print("تقرير الجودة (محاكاة):")
print(f"  متوسط نسبة %GC: {report['gc_content']:.1f}")
print(f"  مواقع منخفضة الجودة: {len(report['low_quality_positions'])} قاعدة")
print(f"  التوصية: {report['recommendation']}")
```

الدوال المعرفة هنا (__init__, __run__) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (FastQCSimulation) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة FastQCSimulation يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

This educational simulation illustrates how quality indicators and processing recommendations are generated. In reality, FastQC uses graphical plots and detailed reports.

هذه المحاكاة التعليمية توضح كيفية توليد مؤشرات الجودة وتوصيات المعالجة. في الواقع، تستخدم FastQC رسوماً بيانية وتقارير تفصيلية.

7. Limits of Quality Control

Quality control is a powerful tool, but it has limits:

7. حدود مراقبة الجودة

مراقبة الجودة أداة قوية، لكن لها حدود:

· لا تعوض عن تصميم تجربة سيء: إذا كانت العينات غير ممثلة أو المكررات غير كافية، فلن تصلح الجودة ذلك.

· It does not compensate for poor experimental design: if samples are unrepresentative or replicates insufficient, quality control will not fix that.

· قد تزيل إشارات حقيقية: في بعض الحالات، قد تكون المناطق منخفضة الجودة تحوي إشارات بيولوجية حقيقية (مثل تعدد الأشكال).

· It may remove real signals: in some cases, low-quality regions may contain real biological signals (e.g., polymorphisms).

· تعتمد على العتبات: اختيار العتبات يؤثر على النتائج، ويجب أن يكون مبرراً.

· It depends on thresholds: choosing thresholds affects results and must be justified.

· قد تخفي مشكلات: قد تمر بعض المشكلات (مثل التلوث الخفيف) دون أن تُكتشف.

· It may hide problems: some issues (such as mild contamination) may go undetected.

ملخص: مراقبة الجودة خطوة ضرورية، لكنها ليست كافية وحدها. يجب دمجها مع تصميم تجربة جيد، وتوثيق دقيق، وتحليل إحصائي مناسب.

Chapter Summary

خلاصة الفصل

Read quality is the foundation of reliable RNA-seq analysis. Quality indicators such as Phred scores, quality distribution, GC content, sequence duplication, and adapters help identify issues. Contamination and technical artifacts can affect results. Preprocessing decisions such as trimming and filtering must be justified and documented, as they affect final results. Trimming or cleaning does not always mean improvement; it may lead to information loss or bias. Documenting what was excluded and why is an essential part of reproducibility. The next chapter will move to Counts and Normalization in RNA-seq analysis.

جودة القراءات هي أساس تحليل RNA-seq الموثوق. مؤشرات الجودة مثل درجات Phred، وتوزيع الجودة، ونسبة GC، وتكرارات التسلسل، والمحولات تساعد في تحديد المشكلات. التلوث والمظاهر التقنية يمكن أن تؤثر على النتائج. قرارات المعالجة المسبقة مثل القص والتصفية يجب أن تكون مبررة وموثقة، لأنها تؤثر على النتائج النهائية. القص أو التنظيف لا يعني دائماً تحسيناً؛ قد يؤدي إلى فقد معلومات أو تحيز. توثيق ما تم استبعاده ولماذا هو جزء أساسي من إعادة الإنتاج. الفصل القادم سينتقل إلى Counts والتطبيع في تحليل RNA-seq.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Read Quality" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "جودة القراءات" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "quality control guarantees error-free data." This is incorrect; quality control detects issues and suggests treatments, but it does not guarantee complete absence of errors. The correct statement is: "quality control is an essential step to identify issues in raw reads and make justified processing decisions, but it does not guarantee error-free data, nor does it compensate for poor experimental design." A simple test: Practical check: "Does quality control guarantee that there are no issues in the data?".

مثال على التجاوز: القول إن "مراقبة الجودة تضمن عدم وجود أخطاء في البيانات". هذا غير صحيح؛ مراقبة الجودة تكتشف المشكلات وتقترح معالجات، لكنها لا تضمن خلواً تاماً من الأخطاء. الصحيح هو: "مراقبة الجودة هي خطوة أساسية لتحديد مشكلات القراءات الخام واتخاذ قرارات معالجة مبررة، لكنها لا تضمن بيانات خالية من الأخطاء، ولا تعوض عن تصميم تجربة سيء." اختبار بسيط: فحص تطبيقي: "هل مراقبة الجودة تضمن عدم وجود أي مشكلة في البيانات؟"ـ

Example: A FastQC report showing low quality at read ends is a computational representation. Claiming that "these low-quality regions contain unimportant biological information" is a biological claim not proven by the quality report alone. The quality report tells us about base quality, but not about biological importance. Practical test: before trimming low-quality regions, Practical check: "Could these regions contain important biological variants?".

مثال: تقرير FastQC يظهر انخفاض الجودة في نهايات القراءات هو تمثيل حسابي. الادعاء بأن "هذه المناطق منخفضة الجودة تحتوي على معلومات بيولوجية غير مهمة" هو ادعاء بيولوجي لا يثبته تقرير الجودة وحده. تقرير الجودة يخبرنا بجودة القواعد، لكنه لا يخبرنا عن أهميتها البيولوجية. اختبار عملي: قبل أن تقص مناطق منخفضة الجودة، فحص تطبيقي: "هل هذه المناطق قد تحتوي على متغيرات بيولوجية مهمة؟"ـ

مثال تطبيقي على مراقبة الجودة:

· المدخلات: ملفات FASTQ (قراءات خام)، معلمات الجودة (مثل عتبة Phred)ـ

· الافتراضات: أن درجات الجودة تعكس دقة القواعد، وأن العتبات المختارة مناسبة للبياناتـ

· المخرجات: تقرير جودة، وتوصيات للمعالجة (مثل القص، التصفية)ـ

· القيود: مراقبة الجودة لا تكتشف جميع المشكلات (مثل التلوث الخفيف)، ولا تحدد الأهمية البيولوجية للقراءاتـ

Applied example of quality control:

اختبار: قبل تطبيق قرارات المعالجة، تحقق من تقارير الجودة لعدة عينات، وتأكد من أن العتبات متسقة.

· Inputs: FASTQ files (raw reads), quality parameters (e.g., Phred threshold).

· Assumptions: that quality scores reflect base accuracy, and that the chosen thresholds are appropriate for the data.

· Outputs: quality report, and processing recommendations (e.g., trimming, filtering).

· Limitations: quality control does not detect all issues (e.g., mild contamination), nor does it determine the biological importance of reads.

Test: before applying processing decisions, check quality reports for multiple samples, and ensure thresholds are consistent.

عند إجراء مراقبة الجودة والمعالجة، يجب تسجيل:

· المصدر: ملفات FASTQ الأصلية، ومعرفات العينات.

· الإصدار: إصدار أدوات مراقبة الجودة (مثل FastQC، MultiQC) والمعالجة (مثل Trimmomatic، cutadapt).

· المعلمات: عتبات الجودة، وطول القص، وأي معاملات أخرى.

· التحويلات: الخطوات المطبقة بالترتيب، وتأثيرها على أعداد القراءات.

When performing quality control and processing, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة نفس خطوات المعالجة والحصول على نفس القراءات المعالجة؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: original FASTQ files, sample identifiers.

· Version: quality control tool versions (e.g., FastQC, MultiQC) and processing tools (e.g., Trimmomatic, cutadapt).

· Parameters: quality thresholds, trim length, and any other parameters.

· Transformations: the steps applied in order, and their impact on read counts.

Test: Practical check: "If I gave this record to a colleague a year later, could they reapply the same processing steps and obtain the same processed reads?" If the answer is no, the documentation is insufficient.

قبل تطبيق أي معالجة على بيانات حقيقية، اختبر مسار العمل على بيانات اصطناعية ذات جودة معروفة. أنشئ قراءات ذات جودة عالية ومنخفضة، وطبّق نفس خطوات المعالجة. تحقق من أن المعالجة تحسن الجودة دون فقدان

معلومات حيوية (في البيانات الاصطناعية المعروفة). إذا نجح الاختبار، فأنت واثق من أن مسار العمل يعمل كما هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس خصائص البيانات وليس خطأ في الإعدادات.

Before applying any processing to real data, test the workflow on synthetic data with known quality. Create reads with high and low quality, and apply the same processing steps. Verify that the processing improves quality without losing vital information (in the known synthetic data). If the test succeeds, the workflow is confident that the workflow works as expected, and any result with real data reflects the data's properties, not an error in the settings.

Chapter 44 — Counts and Normalization

الفصل 44 — Counts والتطبيع

الفكرة المركزية: library size, counts, normalization, assumptions.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Counts represent observations assigned to genes or features under a particular workflow, but they are affected by library size and data properties. Normalization addresses some systematic differences to make comparisons more meaningful; it does not turn the data into biological truth independent of assumptions.

العد يمثل كمية ملاحظات مرتبطة بجينات أو ميزات تحت سير عمل معين، لكنه يتأثر بحجم المكتبة وخصائص البيانات. التطبيع يعالج بعض الاختلافات المنهجية حتى تصبح المقارنة أكثر عدلاً، لكنه لا يحول البيانات إلى حقيقة بيولوجية مستقلة عن الافتراضات.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Raw Counts: What Are They and What Are Their Limits?

1. الأعداد الخام (Raw Counts): ما هي وما حدودها؟

After alignment, we obtain the number of reads that aligned to each gene in each sample. This is the "raw count." But raw counts do not directly reflect expression levels, because they are affected by multiple factors:

بعد المحاذاة، نحصل على عدد القراءات التي حُاذيت لكل جين في كل عينة. هذا هو "العدد الخام" (raw count).

لكن الأعداد الخام لا تعكس مباشرة مستوى التعبير، لأنها تتأثر بعوامل متعددة:

عمق التسلسل (Sequencing Depth): العينات التي سُلسلت بعمق أكبر (أي عدد قراءات أكبر) سيكون لديها أعداد أكبر لجميع الجينات، حتى لو كان التعبير الحقيقي هو نفسه. تخيل أن يوجد في المثال عينتين، إحداهما سُسلت بعمق 10 ملايين قراءة والأخرى بعمق 20 مليون قراءة. جميع الأعداد في العينة الثانية ستكون ضعف الأولى تقريباً، حتى لو كان التعبير متطابقاً.

Sequencing Depth: Samples sequenced at greater depth (i.e., more reads) will have higher counts for all genes, even if the true expression is the same. Imagine two samples, one sequenced at 10 million reads and the other at 20 million reads. All counts in the second sample will be roughly double the first, even if expression is identical.

طول الجين (Gene Length): الجينات الأطول تنتج عدداً أكبر من القراءات، لأنها توفر مساحة أكبر للمحاذاة. جين بطول 1000 قاعدة سيحصل على ضعف قراءات جين بطول 500 قاعدة، حتى لو كان التعبير متساوياً.

Gene Length: Longer genes produce more reads because they provide more space for alignment. A 1000-base gene will get twice the reads of a 500-base gene, even if expression is equal.

تأثيرات تقنية أخرى: مثل كفاءة إعداد المكتبة، وتحيز تضخيم PCR، وتأثيرات الدفعات.

Other Technical Effects: such as library preparation efficiency, PCR amplification bias, and batch effects.

```
# مثال تعليمي: تأثير عمق التسلسل
sample1_counts = [100, 200, 300] # عمق تسلسل: 10 مليون
sample2_counts = [200, 400, 600] # عمق تسلسل: 20 مليون (ضعف الأول)
# الأعداد الخام تبدو وكأن التعبير في العينة الثانية ضعف الأولى
# لكن التعبير الحقيقي قد يكون متساوياً (كل الأعداد تضاعفت بنفس النسبة)
```

Therefore, raw counts cannot be used directly to compare expression between samples.

2. Why Do We Need Normalization؟

لذلك، لا يمكن استخدام الأعداد الخام مباشرة لمقارنة التعبير بين العينات.

Normalization is the process of correcting raw counts to remove the effect of technical factors, so that comparisons between samples become biologically meaningful. Normalization is not optional; it is a mandatory step in any reliable RNA-seq analysis. Without normalization, you may detect false differences (false positives) or miss real differences (false negatives).

2. لماذا نحتاج إلى التطبيع؟

التطبيع هو عملية تصحيح الأعداد الخام لإزالة تأثير العوامل التقنية، بحيث تصبح المقارنات بين العينات ذات معنى بيولوجي. التطبيع ليس خطوة اختيارية؛ هو خطوة إلزامية في أي تحليل RNA-seq موثوق. بدون تطبيع، قد تكتشف اختلافات وهمية (إيجابيات كاذبة) أو تفوت اختلافات حقيقية (سلبيات كاذبة).

أهداف التطبيع:

1. تصحيح اختلافات عمق التسلسل.

2. تصحيح اختلافات أطوال الجينات (إذا لزم الأمر).

3. جعل المقارنات بين العينات ذات معنى.

4. تمكين النمذجة الإحصائية.

But normalization is not just a "button" you press. Each normalization method carries assumptions, and the choice of method affects results. The choice of normalization method should be based on experimental design, data distribution, and the biological question.

3. Common Normalization Methods

لكن التطبيع ليس مجرد "زر" تضغط عليه. كل طريقة تطبيع تحمل افتراضات، واختيار الطريقة يؤثر على النتائج. يجب أن يكون اختيار طريقة التطبيع مستنداً إلى تصميم التجربة، وتوزيع البيانات، والسؤال البيولوجي.

There are several methods for normalizing RNA-seq data, differing in their assumptions and complexity. Some of the most common methods:

3. طرق التطبيع الشائعة

RPKM (Reads Per Kilobase per Million).

هناك عدة طرق لتطبيع بيانات RNA-seq، تختلف في افتراضاتها وتعقيدها. بعض الطرق الأكثر شيوعاً:

تحسب RPKM كالتالي:

· تقسيم عدد القراءات على طول الجين (بالكيلوبايت).

RPKM is calculated as:

· ثم تقسيم الناتج على إجمالي عدد القراءات (بالملايين) في العينة.

· Divide the read count by the gene length (in kilobases).

RPKM يصحح لاختلافات طول الجين وعمق التسلسل. لكنه يفترض أن إجمالي عدد القراءات هو مقياس جيد لحجم المكتبة، وهو افتراض قد لا ينطبق دائماً.

· Then divide the result by the total number of reads (in millions) in the sample.

RPKM corrects for gene length and sequencing depth differences. However, it assumes that the total number of reads is a good measure of library size, an assumption that may not always hold.

TPM (Transcripts Per Million)

TPM is calculated similarly, but divides by the total number of reads after length correction. TPM gives more consistent results for between-sample comparisons.

TPM تحسب بالمثل، لكنها تقسم على إجمالي عدد القراءات بعد تصحيح الطول. TPM تعطي نتائج أكثر اتساقاً للمقارنات بين العينات.

This method, used in the DESeq2 package, calculates a normalization factor for each sample based on the median of ratios between the sample and a reference sample. It assumes that most genes do not change significantly between samples. This method is robust against outliers.

DESeq2's Median-of-Ratios (معدل النسب).

TMM (Trimmed Mean of M-values)

هذه الطريقة، المستخدمة في حزمة DESeq2، تعتمد على حساب عامل تطبيع لكل عينة بناءً على وسيط النسب بين العينة وعينة مرجعية. تفترض أن معظم الجينات لا تتغير بشكل كبير بين العينات. هذه الطريقة قوية ضد القيم المتطرفة.

Similar to DESeq2, used in the edgeR package. Calculates a trimmed mean of log-ratios between samples.

مشابهة لـ DESeq2، تستخدم في حزمة edgeR. تحسب متوسطاً مقطوعاً للنسب اللوغاريتمية بين العينات.

```
def normalize_rpkm(counts, gene_lengths, total_reads):
    """محاكاة تطبيع RPKM."""
    rpkm = []
    for gene_counts, length in zip(counts, gene_lengths):
        rpkm_gene = []
        for count, total in zip(gene_counts, total_reads):
            rpkm_gene.append((count / (length / 1000)) / (total / 1e6))
        rpkm.append(rpkm_gene)
    return np.array(rpkm)

# محاكاة تعليمية لطرق التطبيع المختلفة
import numpy as np
def normalize_deseq2(counts):
    """محاكاة بسيطة لتطبيع DESeq2 (معدل النسب)."""
```

```
محاكاة: حساب وسيط النسب #
reference = np.median(counts, axis=1)  # وسيط عبر العينات
factors = []
for sample in range(counts.shape[1]):
    ratios = counts[:, sample] / reference
    استبعاد الجينات ذات الأعداد الصفرية #
    valid_ratios = ratios[ratios > 0]
    factors.append(np.median(valid_ratios) if len(valid_ratios) > 0 else 1)
  normalized = counts / factors
  return normalized
rpkm = normalize_rpkm(counts, gene_lengths, total_reads)
deseq = normalize_deseq2(counts)
#مثال تعليمي
counts = np.array([
    [100, 200, 150],  # جين 1
    [50, 100, 75],    # جين 2
    [200, 400, 300]   # جين 3
])
gene_lengths = [1000, 500, 2000]  #أطوال الجينات
total_reads = [1e6, 2e6, 1.5e6]  #عمق تسلسل مختلف
print("RPKM (الجين 1):", rpkm[0])
print("DESeq2 تطبيع (الجين 1):", deseq[0])
```

This educational simulation shows how different normalization methods affect results.

4. Connecting Normalization to the Statistical Model

هذه المحاكاة التعليمية تظهر كيف تؤثر طرق التطبيع المختلفة على النتائج.

Normalization is not separate from statistical modeling. In fact, normalization is part of the statistical model itself. In DESeq2, for example, normalization (median-of-ratios) is integrated into the statistical model that assumes the data follows a Negative Binomial distribution. Choosing an inappropriate normalization method may violate model assumptions, leading to incorrect statistical results.

4. ربط التطبيع بالنموذج الإحصائي

التطبيع ليس خطوة منفصلة عن النمذجة الإحصائية. في الواقع، التطبيع هو جزء من النموذج الإحصائي نفسه. في DESeq2، على سبيل المثال، التطبيع (معدل النسب) مدمج في النموذج الإحصائي الذي يفترض أن البيانات تتبع توزيعاً سلبياً ذا حدين (Negative Binomial). اختيار طريقة تطبيع غير مناسبة قد يؤدي إلى انتهاك افتراضات النموذج، وبالتالي نتائج إحصائية غير صحيحة.

مبدأ: طريقة التطبيع يجب أن تتوافق مع النموذج الإحصائي المستخدم.

- RPKM/TPM: مناسبة للتحليلات القائمة على الارتباط أو التجميع.

- DESeq2/edgeR: مناسبة لتحليل التعبير التفاضلي (تحتوي التطبيع في النموذج).

Therefore, the choice of normalization method should be part of the analytical design, not a random decision.

5. Normalization Assumptions and Their Impact

لذلك، يجب أن يكون اختيار طريقة التطبيع جزءاً من التصميم التحليلي، وليس قراراً عشوائياً.

Each normalization method carries assumptions. These assumptions must be understood because they affect results:

5. افتراضات التطبيع وتأثيرها

كل طريقة تطبيع تحمل افتراضات. يجب فهم هذه الافتراضات لأنها تؤثر على النتائج:

افتراض RPKM/TPM:

· أن إجمالي عدد القراءات يعكس حجم المكتبة بشكل موثوق.

· أن توزيع التعبير متجانس بين العينات.

افتراض DESeq2/edgeR:

· هذا قد لا ينطبق إذا كانت هناك اختلافات كبيرة في التعبير بين العينات (مثل عينة مريضة مقابل سليمة).

· أن معظم الجينات لا تختلف بشكل كبير بين العينات.

RPKM/TPM Assumption:

· هذا ينطبق عادةً في تجارب التعبير التفاضلي.

· That the total number of reads reliably reflects library size.

· قوي ضد القيم المتطرفة.

· That expression distribution is homogeneous between samples.

· This may not hold if there are large expression differences between samples (e.g., diseased vs. healthy).

DESeq2/edgeR Assumption:

· That most genes do not differ significantly between samples.

· This typically holds in differential expression experiments.

· Robust against outliers.

مثال: إذا كان 50% من الجينات تتغير بشكل كبير بين عينتين، فإن افتراض DESeq2 (معظم الجينات لا تتغير) ينتهك، وقد تكون النتائج غير موثوقة.

The choice of normalization method should be justified based on experimental design and data characteristics.

6. Normalization and Reproducibility

اختيار طريقة التطبيع يجب أن يكون مبرراً بناءً على تصميم التجربة وخصائص البيانات.

Normalization is a sensitive part of RNA-seq analysis and must be documented precisely. If the normalization method changes, results may change significantly. Therefore, the following must be recorded:

6. التطبيع وقابلية إعادة الإنتاج

التطبيع هو جزء حساس من تحليل RNA-seq، ويجب توثيقه بدقة. إذا تغيرت طريقة التطبيع، قد تتغير النتائج بشكل كبير. لذلك، يجب تسجيل:

· طريقة التطبيع المستخدمة (مثل DESeq2، TPM، RPKM).

· Normalization method used (e.g., RPKM, TPM, DESeq2).

· معلمات التطبيع (مثل العتبات، العينات المرجعية).

· Normalization parameters (e.g., thresholds, reference samples).

· إصدار الأداة المستخدمة (مثل DESeq2 1.40.0).

· Tool version used (e.g., DESeq2 1.40.0).

· مبرر الاختيار: لماذا اختيرت هذه الطريقة وليس غيرها؟

· Justification: why was this method chosen over others?

```
# مثال: توثيق التطبيع
normalization_log = {
    "method": "DESeq2 median-of-ratios",
    "package": "DESeq2",
    "version": "1.40.0",
    "reference_samples": ["sample_01", "sample_02"],
    "justification": "مناسب لتحليل التعبير التفاضلي مع افتراض أن معظم الجينات لا تتغير",
    "date": "2025-01-15"
```

```
}
```

This documentation is an essential part of reproducibility.

7. Practical Application: Impact of Normalization on Results

هذا التوثيق هو جزء أساسي من إعادة الإنتاج.

Let us see how the choice of normalization method affects results in a simple example.

7. التطبيق العملي: تأثير التطبيع على النتائج

دعنا نرى كيف يؤثر اختيار طريقة التطبيع على النتائج في مثال بسيط.

```
#محاكاة تعليمية: تأثير التطبيع على التعبير التفاضلي
import numpy as np
from scipy import stats
# بيانات محاكاة: 1000 جين، 6 عينات (3 سليمة، 3 مريضة)
np.random.seed(42)
```

n_genes = 1000

n_samples = 6

توليد أعداد قراءات خام (محاكاة)

counts = np.random.poisson(lam=100, size=(n_genes, n_samples))

جعل 50 جيناً مختلفة بين الحالتين

for i in range(50):

```
    counts[i, 3:] = counts[i, 3:] * 2  # زيادة في العينات المريضة
```

```
# بيانات الحالة
condition = np.array([0, 0, 0, 1, 1, 1])  # 0 = سليم، 1 = مريض
# تطبيع DESeq2 (محاكاة بسيطة)
reference = np.median(counts, axis=1)
factors = []
for sample in range(n_samples):
    ratios = counts[:, sample] / reference
    valid_ratios = ratios[ratios > 0]
```

deseq_norm = counts / factors

تطبيع RPKM (محاكاة)

total_reads = np.sum(counts, axis=0) # مجموع القراءات لكل عينة

rpkm = counts / total_reads * 1e6

حساب التعبير التفاضلي باستخدام t-test على البيانات المطبعة

def differential_expression(data, condition):

p_values = []

for i in range(data.shape[0]):

group1 = data[i, condition == 0]
group2 = data[i, condition == 1]

_, p = stats.ttest_ind(group1, group2)

```
    p_values.append(p)
  return np.array(p_values)
p_rpkm = differential_expression(rpkm, condition)
p_deseq = differential_expression(deseq_norm, condition)
# عدد الجينات ذات p < 0.05
sig_rpkm = np.sum(p_rpkm < 0.05)
sig_deseq = np.sum(p_deseq < 0.05)
print(f"الجينات المهمة (RPKM): {sig_rpkm}")
print(f"الجينات المهمة (DESeq2): {sig_deseq}")
print(f"50 :الجينات المهمة الحقيقية")
```

This educational simulation shows that the choice of normalization method affects the number of genes detected as differentially expressed.

8. Limits of Normalization

هذه المحاكاة التعليمية توضح أن اختيار طريقة التطبيع يؤثر على عدد الجينات المكتشفة كذات تعبير تفاضلي.

Normalization is a powerful tool, but it has limits:

8. حدود التطبيع

التطبيع أداة قوية، لكن له حدود:

· لا يصحح تصميم تجربة سيء: إذا كانت العينات غير ممثلة أو المكررات غير كافية، فلن يصلح التطبيع ذلك.

· It does not fix poor experimental design: if samples are unrepresentative or replicates insufficient, normalization will not fix that.

· يفترض نماذج معينة: إذا كانت البيانات لا تتبع افتراضات النموذج، فقد تكون النتائج مضللة.

· It assumes specific models: if the data does not follow model assumptions, results may be misleading.

· قد يخفي إشارات حقيقية: في بعض الحالات، قد يؤدي التطبيع إلى إزالة اختلافات بيولوجية حقيقية.

· It may hide real signals: in some cases, normalization may remove real biological differences.

· يعتمد على جودة البيانات: إذا كانت القراءات منخفضة الجودة، فالتطبيع لن يعوض ذلك.

· It depends on data quality: if reads are low quality, normalization will not compensate.

ملخص: التطبيع خطوة ضرورية، لكنها ليست حلاً سحرياً. يجب أن يكون جزءاً من تصميم تحليلي متكامل.

Chapter Summary

خلاصة الفصل

Raw counts from RNA-seq are affected by sequencing depth, gene length, and other technical factors, and cannot be compared directly between samples. Normalization is the process of correcting these counts to make comparisons meaningful. There are multiple normalization methods, such as RPKM, TPM, DESeq2 (median-of-ratios), and TMM. Each method carries assumptions, and the choice of method affects results. Normalization is linked to the statistical model, and the choice of method should be justified based on experimental design and data characteristics. Documenting the normalization method and its parameters is an essential part of reproducibility. Normalization is a powerful tool, but it has limits, and does not compensate for poor experimental design or low-quality data. The next chapter will move to Differential Expression as a final step in RNA-seq analysis.

الأعداد الخام من RNA-seq تتأثر بعمق التسلسل وطول الجين وعوامل تقنية أخرى، ولا يمكن مقارنتها مباشرة بين العينات. التطبيع هو عملية تصحيح هذه الأعداد لجعل المقارنات ذات معنى. هناك طرق تطبيع متعددة، مثل RPKM، TPM، DESeq2 (معدل النسب)، و TMM. كل طريقة تحمل افتراضات، واختيار الطريقة يؤثر على النتائج. التطبيع مرتبط بالنموذج الإحصائي، ويجب أن يكون اختيار الطريقة مبرراً بناءً على تصميم التجربة وخصائص البيانات. توثيق طريقة التطبيع ومعاملاتها هو جزء أساسي من إعادة الإنتاج. التطبيع أداة قوية، لكن له حدود، ولا يعوض عن تصميم تجربة سيء أو بيانات منخفضة الجودة. الفصل القادم سينتقل إلى التعبير التفاضلي (Differential Expression) كخطوة أخيرة في تحليل RNA-seq.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Counts and Normalization" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "Counts والتطبيع" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "normalization removes all technical effects and reveals true expression." This is incorrect; normalization corrects some technical effects, but does not eliminate them completely and depends on assumptions. The correct statement is: "normalization is the process of correcting raw counts to reduce the effect of technical factors, making between-sample comparisons more accurate, but it depends on assumptions and may not eliminate all effects." A simple test: Practical check: "Is this normalization method appropriate for my experimental design and data?".

مثال على التجاوز: القول إن "التطبيع يزيل جميع التأثيرات التقنية ويظهر التعبير الحقيقي". هذا غير صحيح؛ التطبيع يصحح بعض التأثيرات التقنية، لكنه لا يزيلها تماماً، ويعتمد على افتراضات. الصحيح هو: "التطبيع هو عملية تصحيح الأعداد الخام لتقليل تأثير العوامل التقنية، مما يجعل المقارنات بين العينات أكثر دقة، لكنه يعتمد على افتراضات وقد لا يزيل جميع التأثيرات." اختبار بسيط: فحص تطبيقي: "هل طريقة التطبيع هذه مناسبة لتصميم تجربتي وبياناتي؟"ـ

Example: Normalized values are a computational representation of corrected data. Claiming that "these values represent the true expression of genes" is a biological claim not proven by normalization alone. Normalization corrects some technical factors, but true expression depends on additional biological factors. Practical test: before inferring true expression from normalized values, Practical check: "Do these values reflect the relative abundance of RNA in the cell, or are they just computational estimates?".

مثال: قيم التطبيع الناتجة هي تمثيل حسابي للبيانات المصححة. الادعاء بأن "هذه القيم تمثل التعبير الحقيقي للجينات" هو ادعاء بيولوجي لا يثبته التطبيع وحده. التطبيع يصحح بعض العوامل التقنية، لكن التعبير الحقيقي يعتمد على عوامل بيولوجية إضافية. اختبار عملي: قبل أن تستنتج تعبيراً حقيقياً من قيم مطبعة، فحص تطبيقي: "هل هذه القيم تعكس الوفرة النسبية للـ RNA في الخلية، أم أنها مجرد تقديرات حسابية؟"ـ

مثال تطبيقي على تطبيع DESeq2:

· المدخلات: أعداد قراءات خام لكل جين في كل عينة، تصميم التجربة (الحالات، المكررات)ـ

· الافتراضات: أن معظم الجينات لا تتغير بشكل كبير بين العينات، وأن البيانات تتبع توزيعاً سلبياً ذا حدينـ

· المخرجات: أعداد مطبعة، وعوامل تطبيعـ

· القيود: الطريقة قد لا تعمل إذا كانت نسبة كبيرة من الجينات مختلفة بين العينات، أو إذا كانت البيانات تحتوي على قيم متطرفة كثيرةـ

Applied example of DESeq2 normalization:

اختبار: قبل استخدام DESeq2، تحقق من توزيع الأعداد الخام، وتأكد من أن افتراض "معظم الجينات لا تتغير" ينطبق على بياناتكـ

· Inputs: raw read counts for each gene in each sample, experimental design (conditions, replicates).

· Assumptions: that most genes do not change significantly between samples, and that the data follows a Negative Binomial distribution.

· Outputs: normalized counts, and normalization factors.

· Limitations: the method may not work if a large proportion of genes differ between samples, or if the data contains many outliers.

Test: before using DESeq2, check the distribution of raw counts, and verify that the "most genes do not change" assumption applies to the project’s data.

عند تطبيق التطبيع، يجب تسجيل:

· المصدر: الأعداد الخام (ملفات counts)، ومعرفات العينات.

· الإصدار: إصدار الأداة المستخدمة (مثل DESeq2، edgeR)، وإصدار R.

· المعلمات: طريقة التطبيع، والعينات المرجعية (إن وجدت)، وأي معاملات أخرى.

· التحويلات: أي خطوات إضافية قبل أو بعد التطبيع.

When applying normalization, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة التطبيع والحصول على نفس القيم المطبعة؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: raw counts files, sample identifiers.

· Version: the tool version used (e.g., DESeq2, edgeR), and R version.

· Parameters: normalization method, reference samples (if any), and any other parameters.

· Transformations: any additional steps before or after normalization.

Test: Practical check: "If I gave this record to a colleague a year later, could they reapply normalization and obtain the same normalized values?" If the answer is no, the documentation is insufficient.

قبل تطبيق أي طريقة تطبيع على بيانات حقيقية، اختبر الطريقة على بيانات اصطناعية ذات خصائص معروفة. أنشئ بيانات تعبير تحتوي على اختلافات بيولوجية معروفة وتأثيرات تقنية محاكاة. طبّق طرق تطبيع مختلفة وقارن النتائج مع الحقيقة المعروفة. هذا يساعدك على فهم تأثير كل طريقة وتحديد الأنسب لبياناتك. إذا نجح الاختبار، فأنت واثق من أن اختيارك لطريقة التطبيع مناسب.

Before applying any normalization method to real data, test the method on synthetic data with known properties. Create expression data containing known biological differences and simulated technical effects. Apply different normalization methods and compare results with the known truth. This helps you understand the impact of each method and identify the most suitable for the

project’s data. If the test succeeds, the workflow is confident that the project’s choice of normalization method is appropriate.

Chapter 45 — Differential Expression

الفصل 45 — التعبير التفاضلي

الفكرة المركزية: null hypothesis, effect size, dispersion, multiple testing.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

التحليل التفاضلي يختبر ما إذا كانت البيانات متوافقة مع اختلاف بين مجموعات تحت نموذج إحصائي محدد. قيمة p أو FDR لا تخبر وحدها عن السبب البيولوجي، ولا عن حجم الأثر العملي. لذلك يجب الجمع بين حجم الأثر، وعدم اليقين، والتصميم، والسياق البيولوجي.

Differential expression tests whether data are consistent with differences between groups under a specified statistical model. A p-value or FDR does not by itself identify a biological cause or practical effect size. Effect size, uncertainty, design, and biological context should be considered together.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is Differential Expression؟

1. ما هو التعبير التفاضلي؟

Differential expression is a statistical comparison of gene expression levels between two (or more) groups of samples. The goal is to identify genes whose expression changes significantly between groups. But "significant" here has two meanings: statistical (different from chance) and biological (large enough to be meaningful).

التعبير التفاضلي هو مقارنة إحصائية بين مستويات التعبير الجيني في مجموعتين (أو أكثر) من العينات. الهدف هو تحديد الجينات التي تغير تعبيرها بشكل ملحوظ بين المجموعات. لكن "ملحوظ" هنا له معنيان: إحصائي (يختلف عن الصدفة) وبيولوجي (كبير بما يكفي ليكون ذا معنى).

السؤال الأساسي:

هل متوسط التعبير لجين X في المجموعة الأولى يساوي متوسط تعبيره في المجموعة الثانية، أم أنهما مختلفان؟

Answering this question requires a statistical model that accounts for natural variation between samples (biological variation) and technical variation.

2. The Null Hypothesis and Testing It

الإجابة على هذا السؤال تتطلب نموذجاً إحصائياً يأخذ في الاعتبار التباين الطبيعي بين العينات (التباين البيولوجي) والتباين التقني.

In differential expression analysis, we start with a null hypothesis: "there is no difference in expression between the two groups for this gene." We then use a statistical test to determine how compatible the data is with this hypothesis. If the data is incompatible with the null hypothesis (i.e., the probability of seeing such data under the null hypothesis is very small), we reject the null hypothesis and conclude that the gene is differentially expressed.

2. الفرضية العدمية (Null Hypothesis) واختبارها

في تحليل التعبير التفاضلي، نبدأ بفرضية عدمية: "لا يوجد فرق في التعبير بين المجموعتين لهذا الجين". ثم نستخدم اختباراً إحصائياً لتحديد مدى توافق البيانات مع هذه الفرضية. إذا كانت البيانات غير متوافقة مع الفرضية العدمية (أي أن احتمال رؤية مثل هذه البيانات تحت الفرضية العدمية صغير جداً)، نرفض الفرضية العدمية ونستنتج أن الجين يُعبّر بشكل تفاضلي.

قيمة p (p-value):

- هي احتمال الحصول على بيانات متطرفة أو أكثر تطرفاً مما لدينا، بافتراض أن الفرضية العدمية صحيحة.

- قيمة p صغيرة (عادة < 0.05) تعني أن البيانات غير متوافقة مع الفرضية العدمية.

- قيمة p ليست احتمال أن الفرضية العدمية صحيحة، ولا احتمال أن الفرضية البديلة صحيحة.

فهم قيمة p:

Understanding p-value:

· قيمة p = 0.01 لا تعني أن هناك احتمال 1% أن الفرضية العدمية صحيحة.

· p-value = 0.01 does not mean there is a 1% chance the null hypothesis is true.

· تعني: إذا كانت الفرضية العدمية صحيحة، فإن احتمال رؤية بيانات بهذا التطرف (أو أكثر) هو 1%.

· It means: if the null hypothesis is true, the probability of seeing data this extreme (or more) is 1%.

· هذا فرق جوهري يجب فهمه لتجنب سوء التفسير.

· This is a fundamental difference that must be understood to avoid misinterpretation.

3. Effect Size

3. حجم الأثر (Effect Size)

The p-value tells you whether there is a difference, but it does not tell you the size of that difference. Effect size is a measure of how much the groups differ. In differential expression analysis, effect size is usually measured by the Log2 Fold Change.

قيمة p تخبرك ما إذا كان هناك فرق، لكنها لا تخبرك بحجم هذا الفرق. حجم الأثر (effect size) هو مقياس لمدى الاختلاف بين المجموعتين. في تحليل التعبير التفاضلي، يُقاس حجم الأثر عادةً بـ نسبة التغير اللوغاريتمية (Log2 Fold Change).

Language / نوع الكود: **Text / Formula**

Log2 Fold Change = log2(متوسط التعبير في المجموعة الأولى / متوسط التعبير في المجموعة الثانية)

- قيمة موجبة: التعبير أعلى في المجموعة الأولى.

- قيمة سالبة: التعبير أعلى في المجموعة الثانية.

- قيمة صفر: لا فرق.

Why do we use logarithms? Because changes in expression are usually multiplicative (e.g., 2x, 4x), and logarithms make these changes linear and comparable.

لماذا نستخدم اللوغاريتم؟ لأن التغيرات في التعبير تكون عادةً مضاعفة (مثل 2x، 4x)، واللوغاريتم يجعل هذه التغيرات خطية وقابلة للمقارنة.

```
# مثال: Log2 Fold Change
mean_group1 = 100
mean_group2 = 200
fold_change = mean_group2 / mean_group1  # = 2
log2_fold_change = np.log2(fold_change)  # = 1
#(2x زيادة) يعني: التعبير في المجموعة الثانية ضعف المجموعة الأولى
```

4. التباين (Dispersion)

4. Dispersion

Dispersion is a measure of how spread out the data is around the mean. In RNA-seq data, dispersion consists of two parts:

التباين هو مقياس لمدى تشتت البيانات حول المتوسط. في بيانات RNA-seq، التباين يتكون من جزأين:

Biological dispersion: natural variation in expression between individuals or biological replicates. Even under the same conditions, expression varies between samples due to natural biological differences.

التباين البيولوجي (Biological dispersion) : الاختلاف الطبيعي في التعبير بين الأفراد أو المكررات البيولوجية.

حتى في نفس الظروف، يختلف التعبير بين العينات بسبب الاختلافات البيولوجية الطبيعية.

Technical dispersion: variation arising from experimental steps (sequencing quality, library preparation, etc.).

التباين التقني (Technical dispersion) : الاختلاف الناتج عن خطوات التجربة (جودة التسلسل، إعداد المكتبة، إلخ).

التباين الكلي = تباين بيولوجي + تباين تقني

In models like DESeq2, the Negative Binomial distribution is used to model dispersion in expression data. This distribution allows separating technical dispersion from biological dispersion.

في نماذج مثل DESeq2، يُستخدم التوزيع السالب ذو الحدين (Negative Binomial) لنمذجة التباين في بيانات التعبير. هذا التوزيع يسمح بفصل التباين التقني عن التباين البيولوجي.

```
# محاكاة: تأثير التباين على اكتشاف الجينات المختلفة
# جين ذو تباين منخفض: التغير بين المجموعتين واضح
# جين ذو تباين مرتفع: التغير قد يكون بسبب الصدفة
```

5. Multiple Testing

In RNA-seq analysis, thousands of genes are tested simultaneously. If you use p-value = 0.05 for each gene, you would expect 5% of genes to appear different by chance (false positives). With 20,000 genes, this means about 1,000 false positives. Therefore, the p-value must be corrected for multiple testing.

5. الاختبارات المتعددة (Multiple Testing)

في تحليل RNA-seq، تختبر آلاف الجينات في وقت واحد. إذا استخدمت قيمة $p = 0.05$ لكل جين، فمن المتوقع أن تجد 5% من الجينات مختلفة بالصدفة (إيجابيات كاذبة). مع 20,000 جين، هذا يعني حوالي 1,000 إيجابية كاذبة. لذلك، يجب تصحيح قيمة p للاختبارات المتعددة.

FDR is the proportion of false positives among the genes detected as differentially expressed. The Benjamini-Hochberg method is the most common for controlling the FDR. The adjusted p-value is often called the q-value.

معدل الاكتشاف الخاطئ (FDR — False Discovery Rate).

FDR هو نسبة الإيجابيات الكاذبة بين الجينات المكتشفة كذات تعبير تفاضلي. طريقة Benjamini-Hochberg هي الأكثر شيوعاً للتحكم في FDR. قيمة p معدلة (adjusted p-value) تسمى غالباً q-value.

مثال:

- اختبار 20,000 جين.

- عتبة $p < 0.05$ بدون تصحيح: ~1,000 جين مكتشف (قد يكون الكثير منها إيجابيات كاذبة).

- تصحيح FDR عند 5%: يضمن أن 5% فقط من الجينات المكتشفة هي إيجابيات كاذبة.

Rule: use adjusted p-values (or q-values) to identify differentially expressed genes, not raw p-values.

القاعدة: استخدم القيم المعدلة (adjusted p-values أو q-values) لتحديد الجينات ذات التعبير التفاضلي، وليس القيم الخام.

6. Statistical Significance vs. Biological Significance

6. الأهمية الإحصائية مقابل الأهمية البيولوجية

A small p-value means the difference is not due to chance, but it does not mean the difference is biologically important. A gene may be statistically significant (very small p-value) but have a very small effect size (e.g., 1.1x change). This change may not be biologically meaningful. Conversely, a gene may not be statistically significant due to large variance, but have a large effect size (e.g., 5x change) and deserve further study.

قيمة p صغيرة تعني أن الفرق ليس بالصدفة، لكنها لا تعني أن الفرق مهم بيولوجياً. جين قد يكون مختلفاً إحصائياً (p صغيرة جداً) لكن حجم الأثر صغير جداً (مثل تغير 1.1x). هذا التغيير قد لا يكون ذا معنى بيولوجي. والعكس صحيح: جين قد لا يكون مختلفاً إحصائياً بسبب التباين الكبير، لكن حجم الأثر كبير (مثل تغير 5x) ويستحق المزيد من الدراسة.

العوامل الأربعة لتقييم الأهمية البيولوجية:

1. حجم الأثر (Log2 Fold Change) — هل هو كبير بما يكفي؟

2. قيمة p المعدلة — هل هو موثوق إحصائياً؟

3. التباين — هل التقدير دقيق؟

4. السياق البيولوجي — هل هذا الجين معروف في هذه الحالة؟

```
# مثال: جينان مختلفان
# جين A: log2FC = 0.5, p_adj = 0.001 (فرق صغير لكن موثوق)
# جين B: log2FC = 2.5, p_adj = 0.08 (فرق كبير لكن غير موثوق)
# أي الجينين يستحق المتابعة؟ يعتمد على السؤال البيولوجي وقيود الدراسة.
```

7. Practical Application: Differential Expression Analysis Using DESeq2 (Simulation)

In practice, we use packages like DESeq2 or edgeR for differential expression analysis. Here is an educational simulation of how DESeq2 works:

7. التطبيق العملي: تحليل التعبير التفاضلي باستخدام DESeq2 (محاكاة)

في الممارسة العملية، نستخدم حزم مثل DESeq2 أو edgeR لتحليل التعبير التفاضلي. هنا محاكاة تعليمية لكيفية عمل DESeq2:

```
class DESeq2Simulation:
def __init__(self, counts, condition):
self.counts = counts
self.condition = condition
self.n_genes, self.n_samples = counts.shape
self.results = []
# محاكاة تعليمية لتحليل التعبير التفاضلي
import numpy as np
from scipy import stats
def run(self):
"""محاكاة تحليل التعبير التفاضلي."""
for i in range(self.n_genes):
gene_counts = self.counts[i, :]
group1 = gene_counts[self.condition == 0]
group2 = gene_counts[self.condition == 1]
# حساب متوسطات المجموعتين
mean1 = np.mean(group1)
```

```
                mean2 = np.mean(group2)
                # حساب Log2 Fold Change
                if mean1 > 0 and mean2 > 0:
                    log2fc = np.log2(mean2 / mean1)
                else:
                    log2fc = 0
                # حساب التباين (محاكاة)
                var1 = np.var(group1)
                var2 = np.var(group2)
                pooled_var = (var1 + var2) / 2
                # اختبار t (محاكاة)
                if pooled_var > 0:
                    t_stat = (mean2 - mean1) / np.sqrt(pooled_var * (1/len(group1) + 1/len(group2)))
                    p_value = 2 * (1 - stats.t.cdf(abs(t_stat), df=len(group1) + len(group2) - 2))
                else:
                    p_value = 1.0
                self.results.append({
                    "gene": i,
                    "log2fc": log2fc,
                    "p_value": p_value,
                    "mean1": mean1,
                    "mean2": mean2
                })
        # تصحيح القيم p (محاكاة Benjamini-Hochberg)
        self.results.sort(key=lambda x: x["p_value"])
        n = len(self.results)
        for i, res in enumerate(self.results):
            res["p_adj"] = res["p_value"] * n / (i + 1)
        self.results.sort(key=lambda x: x["gene"])
        return self.results
    def get_significant_genes(self, alpha=0.05, lfc_threshold=1):
        """استخراج الجينات ذات التعبير التفاضلي."""
        sig = []
```

for res in self.results:
if res["p_adj"] < alpha and abs(res["log2fc"]) > lfc_threshold:

```
            sig.append(res)
        return sig
# محاكاة بيانات
np.random.seed(42)
```

n_genes = 1000

n_samples = 12 # 6 سليمة، 6 مريضة

توليد أعداد قراءات

counts = np.random.poisson(lam=50, size=(n_genes, n_samples))
condition = np.array([0]*6 + [1]*6)

إضافة 50 جيناً مختلفاً بين الحالتين

for i in range(50):

```
    counts[i, 6:] = counts[i, 6:] * 2
```

```
# تحليل التعبير التفاضلي
deseq = DESeq2Simulation(counts, condition)
results = deseq.run()
significant = deseq.get_significant_genes(alpha=0.05, lfc_threshold=1)
```

print(f"عدد الجينات المكتشفة كذات تعبير تفاضلي: {len(significant)}")

```
print(f"الجينات الحقيقية المختلفة: 50")
print(f"أول 5 جينات مكتشفة:")
for res in significant[:5]:
    print(f"  جين {res['gene']}: log2FC = {res['log2fc']:.2f}, p_adj = {res['p_adj']:.4f}")
```

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات.

وجود الفئة DESeq2Simulation يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها،

بدل تكرار المنطق في كل جزء من خط التحليل.

This educational simulation illustrates the basic principles of differential expression analysis. In reality, DESeq2 uses more complex models and a Negative Binomial distribution.

هذه المحاكاة التعليمية توضح المبادئ الأساسية لتحليل التعبير التفاضلي. في الواقع، DESeq2 يستخدم نماذج أكثر

تعقيداً وتوزيعاً سلبياً ذا حدين.

8. Limits of Differential Expression Analysis

Differential expression analysis is a powerful tool, but it has limits:

8. حدود تحليل التعبير التفاضلي

تحليل التعبير التفاضلي أداة قوية، لكن له حدود:

· يعتمد على جودة البيانات: إذا كانت القراءات منخفضة الجودة أو التطبيع غير مناسب، فالنتائج غير موثوقة.

· يعتمد على التصميم التجريبي: المكررات غير الكافية أو العينات غير الممثلة تقلل القدرة على اكتشاف الجينات المختلفة.

· افتراضات النموذج: إذا كانت البيانات لا تتبع افتراضات النموذج، قد تكون النتائج مضللة.

· Depends on data quality: if reads are low quality or normalization inappropriate, results are unreliable.

· الارتباط لا يساوي السببية: تغير التعبير لا يعني أن الجين يسبب الظاهرة.

· Depends on experimental design: insufficient replicates or unrepresentative samples reduce the power to detect differential genes.

· الأهمية البيولوجية تحتاج إلى سياق: قيمة p صغيرة لا تعني أهمية بيولوجية.

· Model assumptions: if the data does not follow model assumptions, results may be misleading.

· Correlation does not equal causation: changes in expression do not mean the gene causes the phenomenon.

· Biological significance needs context: a small p-value does not mean biological importance.

ملخص: تحليل التعبير التفاضلي هو أداة لتوليد الفرضيات، وليس أداة لإثبات السببية.

Chapter Summary

خلاصة الفصل

Differential expression analysis is a statistical process to identify genes whose expression changes between groups of samples. It relies on a null hypothesis (no difference), and a statistical test (such as t-test or models based on the Negative Binomial distribution). The p-value tells us about compatibility with the null hypothesis, but it does not tell us about effect size. Effect size (Log2 Fold Change) tells us how large the difference is. Correction for multiple testing (FDR) is necessary to avoid false positives. Statistical significance does not equal biological significance; effect size, dispersion, and biological context must be evaluated together. The next chapter will move to PCA and Clustering as exploratory tools for expression data.

تحليل التعبير التفاضلي هو عملية إحصائية لتحديد الجينات التي يتغير تعبيرها بين مجموعات من العينات. يعتمد على فرضية عدمية (لا فرق)، واختبار إحصائي (مثل اختبار t أو النماذج المعتمدة على التوزيع السالب ذي الحدين).

قيمة p تخبرنا عن التوافق مع الفرضية العدمية، لكنها لا تخبرنا بحجم الأثر. حجم الأثر (Log2 Fold Change) يخبرنا بمدى الاختلاف. التصحيح للاختبارات المتعددة (FDR) ضروري لتجنب الإيجابيات الكاذبة. الأهمية الإحصائية لا تساوي الأهمية البيولوجية؛ يجب تقييم حجم الأثر، والتباين، والسياق البيولوجي معاً. الفصل القادم سينتقل إلى PCA و Clustering كأدوات استكشافية لبيانات التعبير.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Differential Expression" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "التعبير التفاضلي" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "a small p-value proves that the gene causes the disease." This is incorrect; a p-value indicates a statistical difference, not causation. The correct statement is: "a small p-value indicates that the expression difference is not due to chance, but proving a causal role requires additional evidence and independent experiments." A simple test: Practical check: "Is this gene only statistically different, or is there evidence that it plays a role in the disease?".

مثال على التجاوز: القول إن "قيمة p الصغيرة تثبت أن الجين يسبب المرض". هذا غير صحيح؛ قيمة p تشير إلى اختلاف إحصائي، وليس سبباً. الصحيح هو: "قيمة p الصغيرة تشير إلى أن الفرق في التعبير ليس بالصدفة، لكن إثبات الدور السببي يحتاج إلى أدلة إضافية وتجارب مستقلة." اختبار بسيط: فحص تطبيقي: "هل هذا الجين مختلف إحصائياً فقط، أم أن هناك أدلة على أنه يلعب دوراً في المرض؟".

Example: A DESeq2 result showing log2FC = 2 and p_adj = 0.001 is a computational representation. Claiming that "this gene is the main cause of the disease" is a biological claim not proven by statistical analysis alone. The analysis identifies genes with changed expression, but proving a causal role requires independent experiments. Practical test: before inferring a biological role from differential expression results, Practical check: "Is there independent evidence (such as gene knockout experiments, or other studies) supporting this role?".

مثال: نتيجة تحليل DESeq2 تعطي log2FC = 2 و p_adj = 0.001 هي تمثيل حسابي. الادعاء بأن "هذا الجين هو المسبب الرئيسي للمرض" هو ادعاء بيولوجي لا يثبته التحليل الإحصائي وحده. التحليل يحدد جينات ذات تغير في التعبير، لكن إثبات الدور السببي يحتاج إلى تجارب مستقلة. اختبار عملي: قبل أن تستنتج دوراً بيولوجياً من نتائج التعبير التفاضلي، فحص تطبيقي: "هل هناك أدلة مستقلة (مثل تجارب تعطيل الجين، أو دراسات أخرى) تدعم هذا الدور؟".

مثال تطبيقي على تحليل التعبير التفاضلي:

· المدخلات: أعداد قراءات مطبعة لكل جين في كل عينة، تصميم التجربة (الحالات، المكررات)، معلمات النموذج.

· الافتراضات: أن البيانات تتبع توزيعاً سلبياً ذا حدين، وأن معظم الجينات لا تتغير بشكل كبير بين العينات، وأن التطبيع صحيح.

· المخرجات: جدول يحتوي على log2FC، p-value، p-adj، لكل جين.

· القيود: التحليل يعتمد على جودة البيانات والتطبيع والنموذج الإحصائي. قد يفوت جينات منخفضة التعبير، وقد تكون هناك إيجابيات كاذبة.

Applied example of differential expression analysis:

اختبار: قبل تحليل التعبير التفاضلي، تحقق من توزيع البيانات، واستخدم رسوم تشخيصية (مثل MA plot، PCA) للكشف عن المشكلات المحتملة.

· Inputs: normalized counts for each gene in each sample, experimental design (conditions, replicates), model parameters.

· Assumptions: that the data follows a Negative Binomial distribution, that most genes do not change significantly between samples, and that normalization is correct.

· Outputs: a table containing log2FC, p-value, p-adj for each gene.

· Limitations: analysis depends on data quality, normalization, and statistical model. Low-expression genes may be missed, and there may be false positives.

Test: before differential expression analysis, check data distribution, and use diagnostic plots (e.g., MA plot, PCA) to detect potential issues.

عند تحليل التعبير التفاضلي، يجب تسجيل:

· المصدر: الأعداد الخام والمطبعة، ومعرفات العينات، وتصميم التجربة.

· المعلمات: طريقة التطبيع، النموذج الإحصائي، العتبات المستخدمة (مثل alpha، lfcThreshold).

· التحويلات: أي خطوات إضافية قبل أو بعد التحليل.

When analyzing differential expression, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة التحليل والحصول على نفس قائمة الجينات المختلفة؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: raw and normalized counts, sample identifiers, experimental design.

· Parameters: normalization method, statistical model, thresholds used (e.g., alpha, lfcThreshold).

· Transformations: any additional steps before or after analysis.

Test: Practical check: "If I gave this record to a colleague a year later, could they re-run the analysis and obtain the same list of significant genes?" If the answer is no, the documentation is insufficient.

قبل تحليل التعبير التفاضلي على بيانات حقيقية، اختبر مسار العمل على بيانات اصطناعية ذات اختلافات معروفة.

أنشئ بيانات تحتوي على جينات مختلفة بشكل معروف بين مجموعتين، وطبّق نفس خطوات التحليل. تحقق من أن

الأداة تكتشف الجينات المختلفة بدقة (حساسية ونوعية). إذا نجح الاختبار، فأنت واثق من أن مسار العمل يعمل كما

هو متوقع، وأي نتيجة مع بيانات حقيقية تعكس خصائص البيانات وليس خطأ في الإعدادات.

Before analyzing differential expression on real data, test the workflow on synthetic data with known differences. Create data containing genes that are known to differ between two groups, and apply the same analysis steps. Verify that the tool accurately detects the different genes (sensitivity and specificity). If the test succeeds, the workflow is confident that the workflow works as expected, and any result with real data reflects the data's properties, not an error in the settings.

Chapter 46 — PCA and Clustering

الفصل 46 — PCA و Clustering

الفكرة المركزية: الاستكشاف، التباين، metadata، batch effects.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

PCA وClustering أدوات استكشافية تساعد على رؤية البنية العامة للعينات. PCA يبحث عن اتجاهات تباين

خطية، بينما clustering يفرض بنية تجميع وفق معيار معين. كلاهما حساس للقياس والقيم الشاذة والاختيارات

المسبقة، لذلك يجب ربط النمط المرئي بالبيانات الوصفية.

PCA and clustering are exploratory tools for viewing broad sample structure. PCA seeks major directions of linear variation, while clustering imposes a grouping structure under a chosen criterion. Both are sensitive to scaling, outliers, and preprocessing choices, so visible patterns should be checked against metadata.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is PCA؟

1. ما هو PCA؟

PCA (Principal Component Analysis) is a statistical technique that reduces the dimensionality of data while retaining as much variation as possible. In RNA-seq data, we have thousands of genes (dimensions) and dozens of samples. PCA finds directions (principal components) that capture the largest amount of variation in the data. The first principal component (PC1) captures the largest direction of variation, the second (PC2) captures the second largest, and so on. By plotting PC1 vs. PC2, we can see how samples cluster in a two-dimensional space.

PCA (Principal Component Analysis) هو أسلوب إحصائي يقلل أبعاد البيانات مع الاحتفاظ بأكبر قدر ممكن من التباين. في بيانات RNA-seq، لدينا آلاف الجينات (الأبعاد) وعشرات العينات. PCA يجد اتجاهات (مكونات رئيسية) تلتقط أكبر قدر من التباين في البيانات. المكون الرئيسي الأول (PC1) يلتقط أكبر اتجاه للتباين، والمكون الثاني (PC2) يلتقط ثاني أكبر اتجاه، وهكذا. برسم PC1 مقابل PC2، يمكننا رؤية كيفية تجمع العينات في فضاء ثنائي الأبعاد.

مثال:

- 10,000 جين × 20 عينة.

- PCA يحولها إلى 20 نقطة في فضاء ثنائي الأبعاد (PC1, PC2).

- نقترب من بعضها إذا كانت العينات متشابهة.

- تبتعد عن بعضها إذا كانت مختلفة.

```
# مثال تعليمي: محاكاة PCA
from sklearn.decomposition import PCA
import numpy as np
import matplotlib.pyplot as plt
# بيانات محاكاة: 1000 جين × 20 عينة
np.random.seed(42)
data = np.random.randn(20, 1000)  # 20 عينة، 1000 جين

# تطبيق PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(data)

# إضافة اختلاف بيولوجي اصطناعي (أول 10 عينات مختلفة عن آخر 10)
data[:10, :] += 2
```

```
# رسم النتائج
plt.figure(figsize=(8, 6))
plt.scatter(pca_result[:10, 0], pca_result[:10, 1], color='blue', label='Group 1')
plt.scatter(pca_result[10:, 0], pca_result[10:, 1], color='red', label='Group 2')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.title('PCA of Simulated Data')
plt.show()
```

This educational example shows how PCA can reveal sample clustering by biological group. But in reality, clustering may be due to technical factors.

2. Interpreting PCA Cautiously

هذا المثال التعليمي يوضح كيف يمكن لـ PCA أن يكشف عن تجمع العينات حسب المجموعة البيولوجية. لكن في الواقع، قد يكون التجميع بسبب عوامل تقنية.

2. تفسير PCA بحذر

PCA is a powerful tool, but its results must be interpreted cautiously:

PCA أداة قوية، لكن يجب تفسير نتائجها بحذر:

التجميع لا يعني السببية: إذا تجمعت العينات حسب الحالة، فهذا يشير إلى أن الحالة تساهم في التباين، لكنه لا يثبت أن الحالة هي السبب الوحيد أو الرئيسي.

Clustering does not mean causation: if samples cluster by condition, it suggests the condition contributes to variation, but it does not prove it is the sole or main cause.

تأثيرات الدفعات (Batch Effects): قد تتجمع العينات حسب دفعة التسلسل، أو المختبر، أو تاريخ الاستخلاص، بدلاً من الحالة البيولوجية. إذا رأيت تجمعاً غير متوقع، تحقق من البيانات الوصفية.

Batch Effects: samples may cluster by sequencing batch, laboratory, or extraction date, rather than by biological condition. If you see unexpected clustering, check the metadata.

المكونات الرئيسية قد لا تكون بيولوجية: PC1 قد يلتقط تأثيراً تقنياً كبيراً (مثل عمق التسلسل)، بينما PC2 قد يلتقط التأثير البيولوجي. من المهم فحص تحميلات المكونات (loadings) لفهم ما تلتقطه كل مكون.

Principal components may not be biological: PC1 may capture a large technical effect (e.g., sequencing depth), while PC2 may capture the biological effect. It is important to examine component loadings to understand what each component captures.

قاعدة: دائماً اربط نتائج PCA بالبيانات الوصفية (metadata).

إذا رأيت تجمعاً، اسأل: هل هذا يتوافق مع الحالة البيولوجية أم مع عامل تقني؟

3. Clustering: Grouping Samples and Genes

Clustering is a technique for grouping similar samples or genes together. There are two main types of clustering used in RNA-seq analysis:

3. Clustering: تجميع العينات والجينات

Clustering هو أسلوب لتجميع العينات أو الجينات المتشابهة معاً. هناك نوعان رئيسيان من التجميع المستخدم في تحليل RNA-seq:

Sample Clustering: to determine whether samples cluster by condition, laboratory, or other factors. This helps detect outliers and batch effects.

تجميع العينات (Sample Clustering): لتحديد ما إذا كانت العينات تتجمع حسب الحالة، أو المختبر، أو عوامل أخرى. هذا يساعد في الكشف عن القيم الشاذة (outliers) وتأثيرات الدفعات.

Gene Clustering: to identify groups of genes with similar expression patterns. These groups may indicate shared biological functions or pathways.

تجميع الجينات (Gene Clustering): لتحديد مجموعات من الجينات ذات أنماط تعبير متشابهة. قد تشير هذه المجموعات إلى وظائف بيولوجية مشتركة أو مسارات مشتركة.

```
# مثال تعليمي: تجميع العينات باستخدام Clustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
# تطبيع البيانات
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
# بيانات محاكاة (نفس البيانات السابقة)
data = np.random.randn(20, 1000)
data[:10, :] += 2
# تطبيق K-means (مجموعتين)
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
clusters = kmeans.fit_predict(data_scaled)
print("تسميات المجموعات:", clusters)
print("العينات 0-9 (المجموعة 1):", clusters[:10])
print("العينات 10-19 (المجموعة 2):", clusters[10:])
```

This educational example shows how clustering can separate two groups. In practice, we typically use Hierarchical Clustering with Heatmaps to visualize expression patterns.

هذا المثال التعليمي يوضح كيف يمكن للتجميع أن يفصل بين المجموعتين. في الواقع، نستخدم عادةً التجميع الهرمي (Hierarchical Clustering) مع الرسوم الحرارية (Heatmaps) لتصور أنماط التعبير.

4. Detecting Batch Effects

Batch effects are technical differences between groups of samples processed at different times, by different people, or in different laboratories. These differences can affect expression and mask the biological signal. Detecting them is an essential step before differential expression analysis.

4. الكشف عن تأثيرات الدفعات (Batch Effects)

تأثيرات الدفعات هي اختلافات تقنية بين مجموعات من العينات تمت معالجتها في أوقات مختلفة، أو بواسطة أشخاص مختلفين، أو في مختبرات مختلفة. يمكن أن تؤثر هذه الاختلافات على التعبير وتخفي الإشارة البيولوجية. الكشف عنها خطوة أساسية قبل تحليل التعبير التفاضلي.

كيف تكتشف تأثيرات الدفعات؟

1. PCA: إذا تجمعت العينات حسب دفعة التسلسل بدلاً من الحالة، فهذا مؤشر قوي.

2. Clustering: إذا تجمعت العينات حسب الدفعة في شجرة التجميع.

3. الرسوم الحرارية: إذا أظهرت أنماطاً غير متوقعة مرتبطة بالدفعة.

```
pca_batch = PCA(n_components=2)
pca_result_batch = pca_batch.fit_transform(data_with_batch)
# محاكاة: تأثير دفعة في PCA

# إضافة تأثير دفعة اصطناعي
data_with_batch = data.copy()
batch_effect = np.random.randn(20, 1) * 0.5
data_with_batch += batch_effect

# إذا كانت العينات تتجمع حسب الدفعة وليس حسب الحالة، فهذا تأثير دفعة.
# If a batch effect is detected, you may need to correct it using methods like ComBat or limma, or include it as a covariate in the statistical model.
```

إذا تم الكشف عن تأثير دفعة، قد تحتاج إلى تصحيحه باستخدام طرق مثل ComBat أو limma، أو تضمينه كمتغير مشترك (covariate) في النموذج الإحصائي.

5. Linking Metadata to Exploratory Plots

5. ربط البيانات الوصفية (Metadata) بالرسوم الاستكشافية

Metadata is information about each sample: condition (patient/healthy), sex, age, sequencing batch, laboratory, extraction date, etc. Without linking this information to exploratory plots, you may misinterpret what you see. In PCA and Clustering plots, points should be colored by different metadata to reveal patterns.

البيانات الوصفية هي معلومات عن كل عينة: الحالة (مريض/سليم)، الجنس، العمر، دفعة التسلسل، المختبر، تاريخ الاستخلاص، إلخ. بدون ربط هذه المعلومات بالرسوم الاستكشافية، قد تسيء تفسير ما تراه. في الرسوم البيانية لـ PCA و Clustering، يجب تلوين النقاط حسب البيانات الوصفية المختلفة للكشف عن الأنماط.

```
# رسم PCA مع تلوين حسب الحالة
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=condition, cmap='coolwarm')
plt.title('PCA colored by Condition')
plt.xlabel('PC1')
plt.ylabel('PC2')
# مثال: تلوين نقاط PCA حسب الحالة والدفعة
# افترض أن لدينا بيانات وصفية: الحالة (0/1) والدفعة (0/1)
condition = np.array([0]*10 + [1]*10)
batch = np.array([0]*5 + [1]*5 + [0]*5 + [1]*5)
# رسم PCA مع تلوين حسب الدفعة
plt.subplot(1, 2, 2)
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=batch, cmap='viridis')
plt.title('PCA colored by Batch')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

This example shows how coloring points by metadata can reveal whether clustering is due to biological condition or a technical factor.

6. Practical Application: PCA and Clustering Analysis for RNA-seq Data

هذا المثال يوضح كيف يمكن لتلوين النقاط بالبيانات الوصفية أن يكشف ما إذا كان التجميع بسبب الحالة البيولوجية أم بسبب عامل تقني.

In practice, we typically use R packages like DESeq2 or edgeR for RNA-seq analysis, and visualization packages like ggplot2 and pheatmap to create PCA and Clustering plots. Here is an educational simulation of the analysis steps:

6. التطبيق العملي: تحليل PCA و Clustering لبيانات RNA-seq

في الممارسة العملية، نستخدم عادةً حزم R مثل DESeq2 أو edgeR لتحليل RNA-seq، ونستخدم حزم التصور مثل ggplot2 و pheatmap لإنشاء رسوم PCA و Clustering. هنا محاكاة تعليمية لخطوات التحليل:

```
# محاكاة تعليمية لتحليل RNA-seq لبيانات Clustering و PCA
import numpy as np
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
class RNAseqExploration:
    def __init__(self, counts, metadata):
        self.counts = counts
        self.metadata = metadata
```

```
        self.scaled_data = None
        self.pca_result = None
    def preprocess(self):
        """تطبيع وتطبيع البيانات."""
        # تحويل لوغاريتمي (محاكاة)
        log_counts = np.log2(self.counts + 1)
        # تطبيع Z-score
        scaler = StandardScaler()
        self.scaled_data = scaler.fit_transform(log_counts.T).T
        return self.scaled_data
    def run_pca(self):
        """تطبيق PCA."""
        pca = PCA(n_components=2)
        self.pca_result = pca.fit_transform(self.scaled_data.T)
        print(f"نسبة التباين المفسر: PC1 = {pca.explained_variance_ratio_[0]:.2f}, PC2 = {pca.explained_variance_ratio_[1]:.2f}")
```

return self.pca_result

def run_clustering(self, n_clusters=2):

"""تطبيق K-means clustering."""

kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)

clusters = kmeans.fit_predict(self.scaled_data.T)

return clusters

def plot_pca(self):

"""رسم PCA مع تلوين حسب البيانات الوصفية."""

if self.pca_result is None:

self.run_pca()

```
plt.figure(figsize=(10, 6))
        scatter = plt.scatter(self.pca_result[:, 0], self.pca_result[:, 1],
                              c=self.metadata['condition'], cmap='coolwarm')
        plt.colorbar(scatter, label='Condition')
        plt.xlabel('PC1')
        plt.ylabel('PC2')
        plt.title('PCA of RNA-seq Data')
        plt.show()
# محاكاة بيانات RNA-seq
np.random.seed(42)
```

n_genes = 1000

n_samples = 20

توليد أعداد قراءات

counts = np.random.poisson(lam=50, size=(n_genes, n_samples))

إضافة اختلاف بيولوجي (أول 10 عينات مختلفة عن آخر 10)

for i in range(100):

counts[i, 10:] = counts[i, 10:] * 2

```
for i in range(100):
```

counts[i, :] += batch_effect

```
# إضافة تأثير دفعة (أول 5 عينات من كل مجموعة في دفعة مختلفة)
batch_effect = np.zeros(n_samples)
batch_effect[:5] = 0.3
batch_effect[5:10] = -0.2
batch_effect[10:15] = 0.4
batch_effect[15:20] = -0.3

# بيانات وصفية
metadata = {
'condition': np.array([0]*10 + [1]*10),
'batch': np.array([0]*5 + [1]*5 + [0]*5 + [1]*5)
}

# تحليل
explorer = RNAseqExploration(counts, metadata)
explorer.preprocess()
explorer.run_pca()
clusters = explorer.run_clustering(n_clusters=2)
print("تسميات المجموعات:", clusters)
print("العينات حسب الحالة:", metadata['condition'])
print("العينات حسب الدفعة:", metadata['batch'])
# رسم PCA
explorer.plot_pca()
```

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات.

وجود الفئة RNAseqExploration يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها،

بدل تكرار المنطق في كل جزء من خط التحليل.

This educational simulation shows how to use PCA and Clustering to explore RNA-seq data, and how metadata can reveal batch effects.

هذه المحاكاة التعليمية توضح كيفية استخدام PCA و Clustering لاستكشاف بيانات RNA-seq، وكيف يمكن

للبيانات الوصفية أن تكشف عن تأثيرات الدفعات.

7. Limits of Exploratory Interpretation

Exploratory interpretation has limits that must be clear:

7. حدود التفسير الاستكشافي

التفسير الاستكشافي له حدود يجب أن تكون واضحة:

- PCA و Clustering ليسا اختبارات فرضيات: لا يمكن استخدامهما لإثبات وجود مجموعات بيولوجية؛ هما أدوات

لتوليد الفرضيات.

· PCA and Clustering are not hypothesis tests: they cannot be used to prove the existence of biological groups; they are tools for generating hypotheses.

· قد يخفيان إشارات حقيقية: إذا كانت التأثيرات التقنية قوية، قد تخفي الإشارة البيولوجية.

· They may hide real signals: if technical effects are strong, they may mask the biological signal.

· قد يظهران إشارات وهمية: إذا كانت البيانات الوصفية غير دقيقة، قد يظهر PCA تجمعاً غير حقيقي.

· They may show spurious signals: if metadata is inaccurate, PCA may show unreal clustering.

· التفسير يعتمد على السياق: ما تراه في الرسم قد يكون حقيقياً أو قد يكون نتيجة لعوامل تقنية. الفهم العميق للبيانات والتجربة هو المفتاح.

· Interpretation depends on context: what you see in the plot may be real or may be due to technical factors. Deep understanding of the data and experiment is the key.

قاعدة: استخدم PCA و Clustering كأدوات استكشافية، وليس كأدلة نهائية.

إذا رأيت شيئاً مثيراً، اسأل: هل هذا متوقع؟ هل يتوافق مع البيانات الوصفية؟ هل يمكن تفسيره بيولوجياً؟

Chapter Summary

خلاصة الفصل

PCA and Clustering are essential exploratory tools for understanding the structure of RNA-seq data. PCA reduces dimensionality and reveals the main directions of variation, while Clustering groups similar samples or genes. PCA and Clustering results must be interpreted cautiously, as they may reflect technical effects (such as batch effects) rather than the biological signal. Linking metadata to exploratory plots is the key to correct interpretation. PCA and Clustering are not hypothesis tests, but tools for generating hypotheses and detecting potential issues. The next chapter will move to the bridge between RNA-seq and biosecurity.

PCA و Clustering هما أدوات استكشافية أساسية لفهم بنية بيانات RNA-seq. PCA يقلل أبعاد البيانات ويكشف الاتجاهات الرئيسية في التباين، بينما Clustering يجمع العينات أو الجينات المتشابهة. يجب تفسير نتائج PCA و Clustering بحذر، لأنها قد تعكس تأثيرات تقنية (مثل batch effects) بدلاً من الإشارة البيولوجية. ربط البيانات الوصفية بالرسوم الاستكشافية هو المفتاح لتفسير صحيح. PCA و Clustering ليسا اختبارات فرضيات،

بل أدوات لتوليد الفرضيات واكتشاف المشكلات المحتملة. الفصل القادم سينتقل إلى الجسر بين RNA-seq والأمن الحيوي.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "PCA and Clustering" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "PCA و Clustering" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "PCA proves the existence of two distinct biological groups." This is incorrect; PCA reveals variation, but it does not prove the existence of biological groups. The correct statement is: "PCA is an exploratory tool that reveals the main directions of variation, and can indicate potential groups, but proving the existence of biological groups requires statistical tests and independent evidence." A simple test: Practical check: "Does this clustering in PCA correspond to metadata, or could it be due to a technical factor?".

مثال على التجاوز: القول إن "PCA يثبت وجود مجموعتين بيولوجيتين متميزتين". هذا غير صحيح؛ PCA يكشف تبايناً، لكنه لا يثبت وجود مجموعات بيولوجية. الصحيح هو: "PCA هو أداة استكشافية تكشف الاتجاهات الرئيسية في التباين، ويمكن أن يشير إلى وجود مجموعات محتملة، لكن إثبات وجود مجموعات بيولوجية يتطلب اختبارات إحصائية وأدلة مستقلة." اختبار بسيط: فحص تطبيقي: "هل هذا التجميع في PCA يتوافق مع البيانات الوصفية، أم أنه قد يكون ناتجاً عن عامل تقني؟".

Example: A PCA plot showing clear clustering of samples by condition is a computational representation. Claiming that "this clustering proves that the condition is the main factor affecting expression" is a biological claim not proven by the plot alone. The plot shows a pattern, but proving that the condition is the main factor requires additional analysis and independent evidence. Practical test: before inferring that the condition is the main factor from a PCA plot, verify that other technical factors (such as batch) do not explain the same clustering.

مثال: رسم PCA يظهر تجمعاً واضحاً للعينات حسب الحالة هو تمثيل حسابي. الادعاء بأن "هذا التجمع يثبت أن الحالة هي العامل الرئيسي المؤثر على التعبير" هو ادعاء بيولوجي لا يثبته الرسم وحده. الرسم يظهر نمطاً، لكن إثبات أن الحالة هي العامل الرئيسي يتطلب تحليلاً إضافياً وأدلة مستقلة. اختبار عملي: قبل أن تستنتج أن الحالة هي العامل الرئيسي من رسم PCA، تأكد من أن عوامل تقنية أخرى (مثل الدفعة) لا تفسر نفس التجميع.

مثال تطبيقي على تحليل PCA و Clustering:

· المدخلات: بيانات مطبعة (أعداد قراءات لكل جين في كل عينة)، بيانات وصفية (الحالة، الدفعة، المختبر).

· الافتراضات: أن التطبيع قد تم بشكل مناسب، وأن البيانات الوصفية دقيقة، وأن اختيار عدد المكونات أو المجموعات مناسب.

· المخرجات: رسوم PCA، رسوم Clustering، تسميات المجموعات.

· القيود: PCA و Clustering لا يحددان الأهمية الإحصائية، ولا يثبتان السببية، وقد يتأثران بالقيم الشاذة والتأثيرات التقنية.

Applied example of PCA and Clustering analysis:

اختبار: قبل تفسير رسوم PCA و Clustering، تحقق من أن البيانات الوصفية دقيقة، واستخدم طرقاً متعددة (مثل تغيير عدد المكونات أو المجموعات) لتقييم استقرار النتائج.

· Inputs: normalized data (counts for each gene in each sample), metadata (condition, batch, laboratory).

· Assumptions: that normalization has been properly performed, that metadata is accurate, and that the choice of components or clusters is appropriate.

· Outputs: PCA plots, Clustering plots, cluster labels.

· Limitations: PCA and Clustering do not determine statistical significance, nor prove causation, and may be affected by outliers and technical effects.

Test: before interpreting PCA and Clustering plots, verify that metadata is accurate, and use multiple methods (e.g., changing the number of components or clusters) to assess result stability.

عند استخدام PCA و Clustering، يجب تسجيل:

· المصدر: البيانات المطبعة (ملفات counts)، والبيانات الوصفية.

· الإصدار: إصدار الأدوات المستخدمة (مثل scikit-learn، DESeq2، ggplot2)، وإصدار R/Python.

· المعلمات: عدد المكونات في PCA، وعدد المجموعات في Clustering، وطريقة التطبيع.

· التحويلات: أي خطوات إضافية (مثل تحويل لوغاريتمي، تطبيع Z-score).

When using PCA and Clustering, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة إنشاء نفس الرسوم والحصول على نفس التجميع؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: normalized data (counts files), metadata.

· Version: the tool versions used (e.g., scikit-learn, DESeq2, ggplot2), and R/Python version.

· Parameters: number of components in PCA, number of clusters in Clustering, and normalization method.

· Transformations: any additional steps (e.g., log transformation, Z-score normalization).

Test: Practical check: "If I gave this record to a colleague a year later, could they recreate the same plots and obtain the same clustering?" If the answer is no, the documentation is insufficient.

قبل تحليل PCA و Clustering على بيانات حقيقية، اختبر المسار على بيانات اصطناعية ذات خصائص معروفة.

أنشئ بيانات تحتوي على مجموعات بيولوجية معروفة وتأثيرات تقنية محاكاة. طبّق PCA و Clustering وتحقق من

أن الأدوات تكتشف المجموعات المتوقعة. إذا نجح الاختبار، فأنت واثق من أن المسار يعمل كما هو متوقع. وعند

تطبيقه على بيانات حقيقية، تذكر أن البيانات الاصطناعية لا تغطي كل التعقيدات الواقعية.

Before applying PCA and Clustering to real data, test the workflow on synthetic data with known properties. Create data containing known biological groups and simulated technical effects. Apply PCA and Clustering and verify that the tools detect the expected groups. If the test succeeds, the workflow is confident that the workflow works as expected. When applying to real data, remember that synthetic data does not cover all real-world complexities.

Chapter 47 — The Bridge Between RNA-seq and Biosecurity

الفصل 47 — الجسر بين RNA-seq والأمن الحيوي

الفكرة المركزية: provenance، decision support، المراجعة، البيانات الحساسة.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

الربط بين RNA-seq والأمن الحيوي هنا منهجي أكثر منه تطبيقي: نفس مبادئ provenance، والتحقق، والفشل

الآمن، وفصل النتائج عن الادعاءات تنطبق على البيانات الجزيئية الحساسة. النظام الجيد لا يعتمد على تحليل واحد،

بل على طبقات تحقق ومراجعة.

The bridge between RNA-seq and biosecurity here is methodological rather than operational: provenance, validation, safe failure, and separation of results from claims apply to sensitive molecular data as well. A robust system uses layers of checks and review rather than relying on a single analysis.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Connecting NGS Outputs to Governance

1. ربط مخرجات NGS بالحوكمة

NGS outputs (such as lists of differentially expressed genes, PCA results, or detected sequences) can be used in multiple security contexts:

مخرجات NGS (مثل قوائم الجينات ذات التعبير التفاضلي، أو نتائج PCA، أو التسلسلات المكتشفة) يمكن استخدامها في سياقات أمنية متعددة:

· الكشف عن التعديلات غير المصرح بها: إذا كانت بيانات RNA-seq تُظهر أنماط تعبير غير متوقعة في كائن حي معدل، فقد يشير ذلك إلى تعديل غير مقصود أو غير مصرح به.

· Detection of unauthorized modifications: if RNA-seq data shows unexpected expression patterns in a modified organism, it may indicate unintended or unauthorized modification.

· تتبع المنشأ (Provenance): يمكن استخدام البيانات لتحديد مصدر العينة، والتحقق من أنها تتطابق مع السجلات المعلنة.

· Provenance tracking: data can be used to identify the sample's source and verify that it matches declared records.

· دعم القرار: يمكن استخدام النتائج كأدلة لدعم قرارات أمنية (مثل الموافقة على تجربة، أو حظرها، أو طلب مراجعة إضافية).

· Decision support: results can be used as evidence to support security decisions (such as approving an experiment, blocking it, or requesting additional review).

· الكشف عن التأثيرات غير المستهدفة: يمكن استخدام تحليل التعبير التفاضلي للكشف عن تغييرات غير متوقعة في التعبير قد تشير إلى تأثيرات غير مرغوب فيها.

· Detection of unintended effects: differential expression analysis can be used to detect unexpected expression changes that may indicate unwanted effects.

مبدأ: مخرجات NGS هي أدلة، وليست أحكاماً نهائية.

يجب أن تُستخدم لدعم القرار البشري، وليس لتحل محله.

2. Automated Classification as Decision Support

In security systems, automated classification (such as machine learning algorithms that classify sequences as dangerous or safe) is used as a first step to narrow the scope of review. But automated classification is not final. It has limits:

2. التصنيف الآلي كدعم للقرار

في الأنظمة الأمنية، يُستخدم التصنيف الآلي (مثل خوارزميات التعلم الآلي التي تصنف التسلسلات على أنها خطيرة أو آمنة) كخطوة أولى لتضييق نطاق المراجعة. لكن التصنيف الآلي ليس نهائياً. له حدود:

· It can be wrong: algorithms are not perfect, and may produce false positives or false negatives.

· قد يخطئ: الخوارزميات ليست مثالية، وقد تنتج إيجابيات كاذبة أو سلبيات كاذبة.

· It lacks context: algorithms do not understand the biological or ethical context of the decision.

· يفتقر إلى السياق: الخوارزميات لا تفهم السياق البيولوجي أو الأخلاقي للقرار.

· It depends on training data: classification quality depends on the quality of the training data.

· يعتمد على البيانات التدريبية: جودة التصنيف تعتمد على جودة البيانات المستخدمة في التدريب.

Therefore, automated classification should be treated as a tool for generating candidates for review, not as a final judgment. All high-consequence or ambiguous cases should undergo human review.

لذلك، يجب التعامل مع التصنيف الآلي كأداة لتوليد المرشحين للمراجعة، وليس كحكم نهائي. جميع الحالات عالية المخاطر أو الملتبسة يجب أن تمر بمراجعة بشرية.

قاعدة: لا تتخذ قراراً أمنياً بناءً على تصنيف آلي فقط.

استخدم التصنيف الآلي لتضييق نطاق المراجعة، ثم دع البشر يتخذون القرار النهائي.

3. Human-in-the-Loop Review

The "human-in-the-loop" principle means that critical decisions are not left entirely to machines. Humans participate in the decision-making process, especially in high-consequence or unclear cases. In the biosecurity context, this means:

3. المراجعة البشرية (Human-in-the-Loop)

مبدأ "الإنسان في الحلقة" (human-in-the-loop) يعني أن القرارات الحاسمة لا تُترك بالكامل للآلات. البشر يشاركون في عملية اتخاذ القرار، خاصة في الحالات عالية المخاطر أو غير الواضحة. في سياق الأمن الحيوي، هذا يعني:

· مراجعة الحالات عالية المخاطر: يجب أن يفحص خبير بشري أي حالة يصنفها النظام على أنها عالية المخاطر.

· Reviewing high-consequence cases: any case classified as high-risk by the system should be examined by a human expert.

· التحقق من الحالات الملتبسة: إذا كان النظام غير متأكد (مثل تشابه حدودي)، يجب أن يراجعها خبير.

· Verifying ambiguous cases: if the system is uncertain (e.g., borderline similarity), an expert should review it.

· تقييم السياق: البشر يمكنهم تقييم السياق البيولوجي والأخلاقي الذي لا يمكن الخوارزميات فهمه.

· Evaluating context: humans can assess the biological and ethical context that algorithms cannot understand.

· المساءلة: القرارات البشرية يمكن تتبعها وتوثيقها، مما يوفر المساءلة.

· Accountability: human decisions can be traced and documented, providing accountability.

```
# مثال تعليمي: نظام مع مراجعة بشرية
class SecurityWorkflow:
    def __init__(self, classifier, review_threshold=0.8):
        self.classifier = classifier
        self.review_threshold = review_threshold
        self.review_queue = []
        self.decisions = []
def process_sample(self, sample_id, data):
    """معالجة عينة وتصنيفها، مع إرسال الحالات الحرجة للمراجعة."""
    # تصنيف آلي
    score = self.classifier.predict(data)
if score >= self.review_threshold:
        # إرسال للمراجعة البشرية
        self.review_queue.append({
                                    "sample_id": sample_id,
                                    "score": score,
                                "status": "pending_review"
                                })
        decision = {
                                    "sample_id": sample_id,
                                    "score": score,
                                "action": "human_review_required",
```

```
                "status": "pending"
            }
        else:
            decision = {
                "sample_id": sample_id,
                "score": score,
                "action": "auto_approved",
                "status": "approved"
            }
        self.decisions.append(decision)
        return decision
    def human_review(self, sample_id, reviewer, decision, rationale):
        """مراجعة بشرية لحالة معلقة."""
        for case in self.review_queue:
            if case["sample_id"] == sample_id:
                case["reviewer"] = reviewer
                case["decision"] = decision
                case["rationale"] = rationale
                case["status"] = "reviewed"
        # تحديث القرار النهائي
        for d in self.decisions:
            if d["sample_id"] == sample_id:
                d["action"] = decision
                d["status"] = "reviewed"
                d["reviewer"] = reviewer
                d["rationale"] = rationale
                break
```

الدوال المعرفة هنا (__init__, process_sample, human_review) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (SecurityWorkflow) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات.

وجود الفئة SecurityWorkflow يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها،

بدل تكرار المنطق في كل جزء من خط التحليل.

This educational example shows a system that combines automated classification and human review.

4. Documentation and Audit Logs

هذا المثال التعليمي يوضح نظاماً يجمع بين التصنيف الآلي والمراجعة البشرية.

In security systems, documentation is not a luxury; it is a necessity. Every step in the workflow, every decision, and every review must be recorded. This allows tracking decisions, identifying errors, improving processes, and providing accountability.

4. التوثيق وسجلات التدقيق

في الأنظمة الأمنية، التوثيق ليس رفاهية؛ إنه ضرورة. يجب تسجيل كل خطوة في سير العمل، وكل قرار، وكل مراجعة.

هذا يسمح بتتبع القرارات، وتحديد الأخطاء، وتحسين العمليات، وتوفير المساءلة.

ما يجب توثيقه في سير العمل الأمني:

1. البيانات المدخلة (مصدرها، تاريخها، هويتها).

2. التصنيف الآلي (النتيجة، العتبات، إصدار النموذج).

3. المراجعة البشرية (من راجع، ماذا قرر، ولماذا).

4. القرار النهائي (موافقة، رفض، تعليق).

5. سجل التدقيق (التسلسل الزمني الكامل).

```
مثال: سجل تدقيق #
audit_log = {
    "sample_id": "SAMPLE_001",
    "input": {
        "source": "unknown",
        "date": "2025-01-15",
        "type": "RNA-seq"
    },
    "automated_classification": {
```

```
"score": 0.85,
"threshold": 0.8,
"model_version": "v1.2",
"date": "2025-01-15T10:00:00"
},
"human_review": {
"reviewer": "expert_01",
"decision": "approved",
"rationale": "التشابه المرتفع يعود إلى منطقة محفوظة، وليس إلى خطر",
"date": "2025-01-15T14:30:00"
},
"final_decision": "approved"
}
```


5. Sensitive Data and Handling It

NGS data, especially that related to pathogens or modified organisms, is considered sensitive data. It must be handled according to strict security procedures:

5. البيانات الحساسة والتعامل معها

بيانات NGS، خاصة تلك المتعلقة بمسببات الأمراض أو الكائنات المعدلة، تعتبر بيانات حساسة. يجب التعامل معها وفق إجراءات أمنية صارمة:

· تقييد الوصول: فقط الأشخاص المخولون يمكنهم الوصول إلى البيانات.

· Access restriction: only authorized personnel can access the data.

· التشفير: يجب تخزين البيانات ونقلها بشكل مشفر.

· Encryption: data must be stored and transmitted in encrypted form.

· سجلات الوصول: تسجيل كل محاولة وصول إلى البيانات.

· Access logs: every access attempt must be recorded.

· التدمير الآمن: حذف البيانات بشكل آمن بعد انتهاء الحاجة إليها.

· Secure destruction: data must be securely deleted after no longer needed.

مبادئ التعامل مع البيانات الحساسة:

1. أقل صلاحية ممكنة: امنح أقل صلاحية ضرورية للقيام بالمهمة.

2. التسجيل: سجل كل وصول وتعديل.

3. التشفير: احم البيانات في التخزين والنقل.

4. المراجعة: راجع سجلات الوصول بانتظام.

6. Limits of Automation in Security Contexts

In security contexts, automation has limits that must be clear:

6. حدود الأتمتة في السياقات الأمنية

في السياقات الأمنية، الأتمتة لها حدود يجب أن تكون واضحة:

· الأتمتة لا تفهم النية: لا يمكن للخوارزمية معرفة ما إذا كان المستخدم لديه نية حسنة أم سيئة.

· Automation does not understand intent: an algorithm cannot know whether a user has good or bad intentions.

· الأتمتة لا تفهم السياق الأخلاقي: لا يمكن للخوارزمية الموازنة بين القيم المتضاربة.

· Automation does not understand ethical context: an algorithm cannot balance conflicting values.

· الأتمتة قابلة للخداع: قد تُخدع الخوارزميات ببيانات مصممة خصيصاً لتجاوزها.

· Automation can be deceived: algorithms can be fooled by data specifically designed to bypass them.

· الأتمتة تحتاج إلى تحديث مستمر: التهديدات تتغير، ويجب تحديث النماذج باستمرار.

· Automation needs continuous updating: threats change, and models must be continuously updated.

ملخص: الأتمتة أداة قوية، لكنها ليست بديلاً عن الحكم البشري، خاصة في السياقات الأمنية.

7. Practical Application: Integrating RNA-seq into a Security Workflow

Let us apply these concepts to a hypothetical security workflow that integrates RNA-seq analysis with human review.

7. التطبيق العملي: دمج RNA-seq في سير عمل أمني

لنطبق هذه المفاهيم على سير عمل أمني وهمي يدمج تحليل RNA-seq مع المراجعة البشرية.

```
# محاكاة تعليمية: سير عمل أمني مع RNA-seq
class SecureRNAseqWorkflow:
    def __init__(self, review_threshold=0.8):
        self.review_threshold = review_threshold
        self.samples = {}
        self.audit_log = []
def add_sample(self, sample_id, metadata):
        """إضافة عينة مع بيانات وصفية."""
        self.samples[sample_id] = {
            "metadata": metadata,
            "status": "pending",
            "analysis": None,
            "review": None
        }
        self._log("add_sample", sample_id, f"تمت إضافة العينة")
    def analyze_rnaseq(self, sample_id, expression_data):
        """تحليل بيانات RNA-seq (محاكاة) وتصنيف المخاطر."""
        if sample_id not in self.samples:
            raise ValueError(f"العينة {sample_id} غير موجودة")
        self.samples[sample_id]["analysis"] = {
            "risk_score": risk_score,
            "classification": "high_risk" if risk_score > self.review_threshold else "low_risk"
        }
        self.samples[sample_id]["status"] = "analyzed"
        self._log("analyze", sample_id, f"تحليل مكتمل. درجة الخطر: {risk_score:.2f}")
        # محاكاة تحليل بسيط (كشف أنماط غير متوقعة)
        # افترض أن التعبير العالي لجينات معينة قد يشير إلى خطر
        risk_score = self._simulate_risk_analysis(expression_data)
        return self.samples[sample_id]["analysis"]
        # إذا كانت عالية المخاطر، إرسال للمراجعة
        if risk_score > self.review_threshold:
            self.samples[sample_id]["status"] = "pending_review"
            self._log("review_required", sample_id, "تم إرسال العينة للمراجعة البشرية")
```

```
def human_review(self, sample_id, reviewer, decision, rationale):
    """مراجعة بشرية للعينة."""
    if sample_id not in self.samples:
        raise ValueError(f"العينة {sample_id} غير موجودة")
    self.samples[sample_id]["review"] = {
        "reviewer": reviewer,
        "decision": decision,
        "rationale": rationale,
        "date": "2025-01-15T14:30:00"
    }
    self.samples[sample_id]["status"] = "reviewed"
    self._log("human_review", sample_id, f"مراجعة بشرية: {decision} - {rationale}")
    return self.samples[sample_id]["review"]

def _simulate_risk_analysis(self, expression_data):
    """محاكاة تحليل المخاطر."""
    # حساب بسيط: ارتفاع التعبير في بعض الجينات يزيد من درجة الخطر
    high_risk_genes = ["GENE_A", "GENE_B", "GENE_C"]
    risk_score = 0
    for gene, value in expression_data.items():
        if gene in high_risk_genes and value > 10:
            risk_score += 0.2
    return min(risk_score, 1.0)  # تطبيع إلى 0-1

def _log(self, action, sample_id, details):
    """تسجيل حدث في سجل التدقيق."""
    self.audit_log.append({
        "timestamp": "2025-01-15T12:00:00",
        "action": action,
        "sample_id": sample_id,
        "details": details
    })
```

```
def get_audit_log(self):
      return self.audit_log
# مثال على الاستخدام
workflow = SecureRNAseqWorkflow(review_threshold=0.6)
# إضافة عينة
workflow.add_sample("SAMPLE_001", {"source": "unknown", "organism": "E. coli"})
# بيانات تعبير محاكاة (محاكاة)
expression_data = {
                                    "GENE_A": 15,
                                    "GENE_B": 3,
                                    "GENE_C": 8,
                                    "GENE_D": 5
                                    }
# تحليل RNA-seq
analysis = workflow.analyze_rnaseq("SAMPLE_001", expression_data)
print(f"تحليل: {analysis}")
# مراجعة بشرية (إذا كانت مطلوبة)
if workflow.samples["SAMPLE_001"]["status"] == "pending_review":
   review = workflow.human_review(
                                    "SAMPLE_001",
                                    "expert_01",
                                    "approved",
"تم التحقق: الجينات المرتفعة تنتمي إلى مسار معروف غير خطير"
                                    )
   print(f"مراجعة: {review}")
# عرض سجل التدقيق
print("\nسجل التدقيق:")
for entry in workflow.get_audit_log():
   print(f"  {entry['action']} - {entry['sample_id']}: {entry['details']}")
```

الدوال المعرفة هنا (__init__, add_sample, analyze_rnaseq, human_review, _simulate_risk_analysis, _log, get_audit_log) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود الأصناف (SecureRNAseqWorkflow) ينقل البرنامج من مجموعة أوامر متفرقة إلى نموذج بيانات له حالة وسلوك. هذا مهم في المعلوماتية الحيوية لأن العينة أو القراءة أو سجل الجودة يمكن أن يحمل مع البيانات القيود التي يجب التحقق منها قبل التحليل. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها.

في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل. في سير العمل، يجب قراءة العلاقة بين المدخلات والمخرجات والموارد والسجل باعتبارها رسمًا سببيًا للتحليل. نجاح خطوة منفردة لا يثبت أن الخط كله قابل لإعادة الإنتاج؛ الإصدار والبيئة والبيانات المرجعية جزء من التعريف الكامل للعملية.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة SecureRNAseqWorkflow يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

This educational simulation shows how to integrate RNA-seq analysis into a security workflow with human review and audit logs.

هذه المحاكاة التعليمية توضح كيفية دمج تحليل RNA-seq في سير عمل أمني مع مراجعة بشرية وسجلات تدقيق.

8. Security Limits in This Model

The educational model has clear security limits:

8. الحدود الأمنية في هذا النموذج

النموذج التعليمي له حدود أمنية واضحة:

· لا يحتوي على بيانات حساسة حقيقية.

· لا يستخدم خوارزميات أمنية تشغيلية.

· It does not contain real sensitive data.

· لا يقدم إرشادات للتجارب عالية المخاطر.

· It does not use operational security algorithms.

· السياسات مبسطة ولا تعكس التعقيدات القانونية أو الأخلاقية.

· It does not provide guidance for high-consequence experiments.

· الهدف هو تعليم المبادئ، وليس تقديم أداة جاهزة.

· Policies are simplified and do not reflect legal or ethical complexities.

· The goal is to teach principles, not to provide a ready-made tool.

Chapter Summary

خلاصة الفصل

RNA-seq and NGS outputs can support defensive workflows in security contexts, but automated classification should be treated as decision support, not as a final decision. High-consequence or ambiguous cases require human review (human-in-the-loop). Documentation and audit logs are the foundation of accountability and reproducibility. Sensitive data must be handled according to strict security procedures. Automation is a powerful tool, but it is not a substitute for human judgment, especially in security contexts. The next chapter will move to Part 12 — Biostatistics.

يمكن استخدام مخرجات RNA-seq و NGS لدعم سير العمل الدفاعي في السياقات الأمنية، لكن يجب التعامل مع التصنيف الآلي كدعم للقرار، وليس كقرار نهائي. الحالات عالية المخاطر أو الملتبسة تتطلب مراجعة بشرية (human-in-the-loop). التوثيق وسجلات التدقيق هي أساس المساءلة وإعادة الإنتاج. البيانات الحساسة يجب التعامل معها وفق إجراءات أمنية صارمة. الأتمتة أداة قوية، لكنها ليست بديلاً عن الحكم البشري، خاصة في السياقات الأمنية. الفصل القادم سينتقل إلى الباب 12 — الإحصاء الحيوي.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "The Bridge Between RNA-seq and Biosecurity" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "الجسر بين RNA-seq والأمن الحيوي" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "RNA-seq analysis can automatically detect all security threats." This is incorrect; analysis is a support tool, not a complete detection system. The correct statement is: "RNA-seq outputs can be used as evidence to support security decisions, but they require human review to handle high-consequence and ambiguous cases, and do not constitute an independent security system." A simple test: Practical check: "Can this analysis replace human judgment in security decisions?".

مثال على التجاوز: القول إن "تحليل RNA-seq يمكنه كشف جميع التهديدات الأمنية تلقائياً". هذا غير صحيح؛ التحليل أداة دعم، وليس نظام كشف كامل. الصحيح هو: "يمكن استخدام مخرجات RNA-seq كأدلة لدعم القرارات الأمنية، لكنها تتطلب مراجعة بشرية للتعامل مع الحالات عالية المخاطر والملتبسة، ولا تشكل نظاماً أمنياً مستقلاً."

اختبار بسيط: فحص تطبيقي: "هل هذا التحليل يمكنه استبدال الحكم البشري في القرارات الأمنية؟"ـ

Example: An RNA-seq result classifying a sample as high-risk is a computational representation. Claiming that "this sample poses a security threat" is a security claim not proven by automated classification alone. Classification generates an indicator, but the final security decision requires context and human review. Practical test: before making a security decision based on RNA-seq analysis, ensure that there is human review and appropriate context.

مثال: نتيجة تحليل RNA-seq تصنف عينة على أنها عالية المخاطر هو تمثيل حسابي. الادعاء بأن "هذه العينة تشكل تهديداً أمنياً" هو ادعاء أمني لا يثبته التصنيف الآلي وحده. التصنيف يولد مؤشراً، لكن القرار الأمني النهائي يتطلب سياقاً ومراجعة بشرية. اختبار عملي: قبل أن تتخذ قراراً أمنياً بناءً على تحليل RNA-seq، تأكد من أن هناك مراجعة بشرية وسياقاً مناسباً.

مثال تطبيقي على سير عمل أمني مع RNA-seq:

· المدخلات: بيانات RNA-seq (مطبعة)، بيانات وصفية، سياسات أمنية، عتبات تصنيف.

· الافتراضات: أن التطبيع والتحليل صحيحان، وأن السياسات تعكس القيم الأمنية، وأن المراجعين مؤهلون.

· المخرجات: تصنيف مخاطر، توصيات (موافقة، مراجعة، رفض)، سجلات تدقيق.

· القيود: التحليل لا يكتشف جميع التهديدات، وقد ينتج إيجابيات أو سلبيات كاذبة، ولا يستبدل الحكم البشري.

Applied example of a security workflow with RNA-seq:

اختبار: قبل الاعتماد على سير عمل أمني، تحقق من سياسات التصنيف، ومؤهلات المراجعين، وسلامة سجلات التدقيق.

· Inputs: RNA-seq data (normalized), metadata, security policies, classification thresholds.

· Assumptions: that normalization and analysis are correct, that policies reflect security values, and that reviewers are qualified.

· Outputs: risk classification, recommendations (approve, review, reject), audit logs.

· Limitations: analysis does not detect all threats, may produce false positives or negatives, and does not replace human judgment.

Test: before relying on a security workflow, verify classification policies, reviewer qualifications, and audit log integrity.

عند استخدام مخرجات RNA-seq في سياق أمني، يجب تسجيل:

· المصدر: البيانات الخام والمطبعة، ومعرفات العينات، والبيانات الوصفية.

· الإصدار: إصدار أدوات التحليل والنماذج الأمنية.

· المعلمات: عتبات التصنيف، والسياسات المطبقة، وأي معاملات أخرى.

· التحويلات: أي خطوات تحليل أو تصفية قبل الاستخدام الأمني.

When using RNA-seq outputs in a security context, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لمراجع بعد عام، هل يستطيع فهم كيف اتخذت القرارات، وما هي البيانات والتحليلات التي استُخدمت؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: raw and normalized data, sample identifiers, metadata.

· Version: analysis tool versions and security model versions.

· Parameters: classification thresholds, policies applied, and any other parameters.

· Transformations: any analysis or filtering steps before security use.

Test: Practical check: "If I gave this record to a reviewer a year later, could they understand how decisions were made, and what data and analyses were used?" If the answer is no, the documentation is insufficient.

قبل تطبيق أي سير عمل أمني يعتمد على RNA-seq على بيانات حقيقية، اختبره على بيانات اصطناعية ذات سيناريوهات معروفة. أنشئ بيانات تحاكي حالات منخفضة وعالية المخاطر، واختبر التصنيف الآلي والمراجعة البشرية. تحقق من أن النظام يتعامل مع كل حالة بشكل صحيح، وأن سجلات التدقيق تسجل كل القرارات. إذا نجح الاختبار، فأنت واثق من أن سير العمل يعمل كما هو متوقع.

Before applying any RNA-seq-based security workflow to real data, test it on synthetic data with known scenarios. Create data simulating low- and high-risk cases, and test automated classification and human review. Verify that the system handles each case correctly, and that audit logs record all decisions. If the test succeeds, the workflow is confident that the workflow works as expected.

Part 12 — Biostatistics

الباب 12 — الإحصاء الحيوي

الفكرة المركزية: المتوسط، الوسيط، التشتت، الرسوم، جودة الوصف.

Chapter 48 — Description Before Inference

الفصل 48 — الوصف قبل الاستدلال

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Descriptive statistics are the first sensible inspection: how values are distributed, where they center, how they vary, and whether outliers exist. Moving to inference before understanding this structure can make a model answer a different question from the one posed by the data.

الإحصاء الوصفي هو أول اختبار معقول للبيانات: كيف تتوزع القيم، وما مركزها، وما مدى تشتتها، وهل توجد قيم شاذة؟ الانتقال إلى الاختبار الاستدلالي قبل فهم هذه البنية قد يجعل النموذج يجيب عن سؤال مختلف عن السؤال الذي تفرضه البيانات.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Why Description Before Inference؟

1. لماذا الوصف قبل الاستدلال؟

Descriptive statistics are the first step in any data analysis. Before testing hypotheses, we must understand the shape of the data: what is the range? What is the distribution? Are there outliers? Is the data symmetric or skewed? Answering these questions determines which statistical tests are appropriate, and whether their assumptions are met. Skipping this step may lead to using inappropriate tests and incorrect conclusions.

الإحصاء الوصفي هو الخطوة الأولى في أي تحليل بيانات. قبل أن نختبر الفرضيات، يجب أن نفهم شكل البيانات: ما هو النطاق؟ ما هو التوزيع؟ هل هناك قيم متطرفة؟ هل البيانات متناظرة أم منحرفة؟ الإجابة على هذه الأسئلة تحدد أي الاختبارات الإحصائية مناسبة، وما إذا كانت افتراضاتها مستوفاة. تخطي هذه الخطوة قد يؤدي إلى استخدام اختبارات غير مناسبة، واستنتاجات خاطئة.

> مبدأ أساسي:
>
> افهم بياناتك قبل أن تختبر فرضياتك.
>
> الوصف أولاً، ثم الاستدلال.

In bioinformatics, this principle is especially important. Gene expression data, for example, is often skewed and contains outliers. Before applying a t-test or ANOVA, the data distribution should be checked and a transformation (such as logarithmic) may be applied to make it more symmetric.

في المعلوماتية الحيوية، هذا المبدأ مهم بشكل خاص. بيانات التعبير الجيني، على سبيل المثال، غالباً ما تكون منحرفة (skewed) وتحتوي على قيم متطرفة. قبل تطبيق اختبار t أو ANOVA، يجب التحقق من توزيع البيانات وربما تطبيق تحويل (مثل لوغاريتمي) لجعلها أكثر تناسقاً.

2. Measures of Central Tendency

2. مقاييس النزعة المركزية (Measures of Central Tendency)

Measures of central tendency describe the "central value" of the data. There are three main measures:

مقاييس النزعة المركزية تصف "القيمة المركزية" للبيانات. هناك ثلاثة مقاييس رئيسية:

The mean is the sum of values divided by their count. The mean is sensitive to outliers. If the workflow has large outliers, the mean will shift toward them. The mean is suitable for symmetric data that approximately follows a normal distribution.

المتوسط (Mean)

هو مجموع القيم مقسوماً على عددها. المتوسط حساس للقيم المتطرفة. إذا كان يوجد في المثال قيم متطرفة كبيرة، فإن المتوسط سينحرف نحوها. المتوسط مناسب للبيانات المتناظرة التي تتبع توزيعاً طبيعياً تقريباً.

الوسيط (Median)

The median is the middle value when data is sorted in ascending order. The median is more resistant to outliers than the mean. If the workflow has outliers, the median gives a better picture of the central value. The median is suitable for skewed data or data containing outliers.

هو القيمة الوسطى عندما تُرتّب البيانات تصاعدياً. الوسيط أكثر مقاومة للقيم المتطرفة من المتوسط. إذا كان يوجد في المثال قيم متطرفة، فالوسيط يعطي صورة أفضل للقيمة المركزية. الوسيط مناسب للبيانات المنحرفة (skewed) أو التي تحتوي على قيم متطرفة.

المنوال (Mode)

The mode is the most frequent value in the data. The mode is useful for categorical data or when you want to know the most common value. In continuous data, there may not be a clear mode.

هو القيمة الأكثر تكراراً في البيانات. المنوال مفيد للبيانات الفئوية (categorical) أو عندما تريد معرفة القيمة الأكثر شيوعاً. في البيانات المستمرة، قد لا يكون هناك منوال واضح.

```
# مثال: مقارنة المتوسط والوسيط
import numpy as np
data_symmetric = np.random.normal(10, 2, 100)  # توزيع طبيعي
data_skewed = np.random.exponential(2, 100)  # توزيع منحرف
print("بيانات متناظرة:")
print(f" المتوسط: {np.mean(data_symmetric):.2f}")
print(f" الوسيط: {np.median(data_symmetric):.2f}")
print("\nبيانات منحرفة:")
print(f" المتوسط: {np.mean(data_skewed):.2f}")
print(f" الوسيط: {np.median(data_skewed):.2f}")
```

In symmetric data, the mean and median are close. In skewed data, they differ, and the median gives a better picture of the central value.

3. Measures of Dispersion

في البيانات المتناظرة، المتوسط والوسيط متقاربان. في البيانات المنحرفة، يختلفان، والوسيط يعطي صورة أفضل للقيمة المركزية.

Measures of dispersion describe how spread out the data is around the central value. There are several main measures:

3. مقاييس التشتت (Measures of Dispersion)

مقاييس التشتت تصف مدى انتشار البيانات حول القيمة المركزية. هناك عدة مقاييس رئيسية:

المدى (Range)

The range is the difference between the largest and smallest values. The range is simple but very sensitive to outliers.

هو الفرق بين أكبر وأصغر قيمة. المدى بسيط لكنه حساس جداً للقيم المتطرفة.

The standard deviation measures the average distance of values from the mean. The standard deviation is suitable for symmetric data. The larger the standard deviation, the more spread out the data.

الانحراف المعياري (Standard Deviation).

المدى الربيعي (Interquartile Range — IQR).

The interquartile range (IQR) is the difference between the third quartile (Q3) and the first quartile (Q1). The IQR measures the spread of the middle 50% of the data and is resistant to outliers. The IQR is suitable for skewed data or data containing outliers.

يقيس متوسط بعد القيم عن المتوسط. الانحراف المعياري مناسب للبيانات المتناظرة. كلما زاد الانحراف المعياري، زاد تشتت البيانات.

هو الفرق بين الربيع الثالث (Q3) والربيع الأول (Q1). IQR يقيس تشتت الـ 50% الوسطى من البيانات، وهو مقاوم للقيم المتطرفة. IQR مناسب للبيانات المنحرفة أو التي تحتوي على قيم متطرفة.

```
# مثال: مقارنة مقاييس التشتت
data = np.random.exponential(2, 100)
print(f"المدى: {np.max(data) - np.min(data):.2f}")
print(f"الانحراف المعياري: {np.std(data):.2f}")
print(f"المدى الربيعي (IQR): {np.percentile(data, 75) - np.percentile(data, 25):.2f}")
```

4. Choosing the Right Measures

Choosing the right measure depends on the distribution shape and data type:

4. اختيار المقاييس المناسبة

اختيار المقياس المناسب يعتمد على شكل التوزيع ونوع البيانات:

نوع البيانات النزعة المركزية التشتت ملاحظات.

Data Type Central Tendency Dispersion Notes.

متناظرة، طبيعية تقريباً المتوسط الانحراف المعياري مناسب للاختبارات البارامترية.

Symmetric, near-normal Mean Standard Deviation Suitable for parametric tests.

منحرفة أو بها قيم متطرفة الوسيط المدى الربيعي (IQR) أكثر مقاومة.

Skewed or with outliers Median Interquartile Range (IQR) More robust.

فئوية المنوال التكرارات النسبية لا يوجد متوسط أو وسيط ذو معنى.

Categorical Mode Relative frequencies No meaningful mean or median.

قاعدة: اختر المقياس الذي يناسب شكل بياناتك، وليس المقياس الأسهل.

5. Inspecting Distributions Before Inference

Before applying any statistical test (such as t-test, ANOVA, or regression models), the data distribution should be inspected. The most common tools for distribution inspection are:

5. فحص التوزيعات قبل الاستدلال

Histograms: show the shape of the distribution. Is it symmetric? Skewed? Multimodal?

قبل تطبيق أي اختبار إحصائي (مثل t-test، ANOVA، أو نماذج الانحدار)، يجب فحص توزيع البيانات. الأدوات الأكثر شيوعاً لفحص التوزيع هي:

Boxplots: show medians, IQR, and outliers. Useful for comparing distributions between groups.

الرسوم البيانية (Histograms): تظهر شكل التوزيع. هل هو متماثل؟ منحرف؟ متعدد القمم؟

Q-Q plots (Quantile-Quantile plots): compare the distribution to a normal distribution. If points fall on a straight line, the data is approximately normal.

Boxplots: تظهر الوسطاء، والمدى الربيعي، والقيم المتطرفة. مفيدة لمقارنة التوزيعات بين مجموعات.

Q-Q plots (Quantile-Quantile plots): تقارن التوزيع بالتوزيع الطبيعي. إذا كانت النقاط تقع على خط مستقيم، فالبيانات طبيعية تقريباً.

```
# مثال: فحص التوزيع باستخدام الرسوم
import matplotlib.pyplot as plt
import seaborn as sns
fig, axes = plt.subplots(2, 3, figsize=(12, 8))

# بيانات محاكاة
data1 = np.random.normal(10, 2, 100)     # طبيعية
data2 = np.random.exponential(2, 100)    # منحرفة
# Histograms
axes[0, 0].hist(data1, bins=20, alpha=0.7)
axes[0, 0].set_title('Normal Data - Histogram')
axes[0, 1].hist(data2, bins=20, alpha=0.7)
axes[0, 1].set_title('Skewed Data - Histogram')

# Boxplots
axes[0, 2].boxplot([data1, data2], labels=['Normal', 'Skewed'])
axes[0, 2].set_title('Boxplots')

# Q-Q plots
from scipy import stats
stats.probplot(data1, dist="norm", plot=axes[1, 0])
axes[1, 0].set_title('Normal Q-Q Plot')

stats.probplot(data2, dist="norm", plot=axes[1, 1])
axes[1, 1].set_title('Skewed Q-Q Plot')
axes[1, 2].axis('off')

plt.tight_layout()
plt.show()
```

These plots help determine whether the data follows a normal distribution, a key assumption for many statistical tests.

6. Practical Application: Describing Gene Expression Data

هذه الرسوم تساعد في تحديد ما إذا كانت البيانات تتبع التوزيع الطبيعي، وهو افتراض أساسي للعديد من الاختبارات الإحصائية.

Let us apply these principles to simulated gene expression data:

6. التطبيق العملي: وصف بيانات تعبير جيني

لنطبق هذه المبادئ على بيانات تعبير جيني محاكاة:

```
# حساب مقاييس وصفية لكل جين
means = np.mean(expression_data, axis=1)
medians = np.median(expression_data, axis=1)
stds = np.std(expression_data, axis=1)
iqrs = np.percentile(expression_data, 75, axis=1) - np.percentile(expression_data, 25, axis=1)
# محاكاة بيانات تعبير جيني (100 جين × 20 عينة)
np.random.seed(42)
expression_data = np.random.exponential(5, (100, 20))
# عرض أول 5 جينات
print("مقاييس وصفية - أول 5 جينات:")
for i in range(5):
    print(f"جين {i}: المتوسط={means[i]:.2f}, الوسيط={medians[i]:.2f}, "
          f"الانحراف المعياري={stds[i]:.2f}, IQR={iqrs[i]:.2f}")
```

```
# رسم توزيع الجين
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
plt.hist(gene_1, bins=15, alpha=0.7)
plt.title('Histogram of Gene 1')
plt.xlabel('Expression')
plt.ylabel('Frequency')
# فحص توزيع جين واحد
gene_1 = expression_data[0, :]
print(f"\nالملخص - جين 1:")
print(f"  المتوسط: {np.mean(gene_1):.2f}")
print(f"  الوسيط: {np.median(gene_1):.2f}")
print(f"  الانحراف المعياري: {np.std(gene_1):.2f}")
print(f"  IQR: {np.percentile(gene_1, 75) - np.percentile(gene_1, 25):.2f}")
plt.subplot(1, 2, 2)
plt.boxplot(gene_1)
plt.title('Boxplot of Gene 1')
plt.ylabel('Expression')
plt.tight_layout()
plt.show()
```

This example shows how to calculate descriptive measures for expression data, and how to inspect their distribution using plots.

7. Limits of Descriptive Statistics

هذا المثال يوضح كيفية حساب مقاييس وصفية لبيانات التعبير، وكيفية فحص توزيعها باستخدام الرسوم.

Descriptive statistics are a powerful tool, but they have limits:

7. حدود الوصف الإحصائي

الإحصاء الوصفي أداة قوية، لكن له حدود:

· لا يخبرك بالسبب: يصف ما هو موجود، لكن لا يشرح لماذا.

· It does not tell you why: it describes what is present, but does not explain why.

· قد يكون مضللاً: إذا كان التوزيع متعدد القمم، قد لا يعكس المتوسط أو الوسيط الصورة الكاملة.

· It can be misleading: if the distribution is multimodal, the mean or median may not reflect the full picture.

· يعتمد على جودة البيانات: إذا كانت البيانات غير دقيقة، فالإحصاء الوصفي سيكون غير دقيق أيضاً.

· It depends on data quality: if the data is inaccurate, the descriptive statistics will also be inaccurate.

· لا يحل محل الرسوم: الرسوم البيانية غالباً ما تكشف أنماطاً لا تظهرها الأرقام وحدها.

· It does not replace plots: plots often reveal patterns that numbers alone do not show.

قاعدة: استخدم الإحصاء الوصفي والرسوم معاً.

الأرقام تعطي ملخصاً، والرسوم تعطي صورة كاملة.

8. From Description to Inference

After describing the data and understanding its distribution, we become ready to move to statistical inference: hypothesis testing, parameter estimation, and model building. But inference depends on description. If the description is wrong or incomplete, the inference will be wrong. Therefore, description is not just a preparatory step; it is the foundation of all subsequent analysis.

8. من الوصف إلى الاستدلال

بعد وصف البيانات وفهم توزيعها، نصبح جاهزين للانتقال إلى الاستدلال الإحصائي: اختبار الفرضيات، تقدير المعلمات، وبناء النماذج. لكن الاستدلال يعتمد على الوصف. إذا كان الوصف خاطئاً أو غير مكتمل، فسيكون الاستدلال خاطئاً. لذلك، الوصف ليس مجرد خطوة تمهيدية؛ إنه أساس كل تحليل لاحق.

مسار التحليل الإحصائي:

1. وصف البيانات (إحصاءات ورسوم).

2. فحص الافتراضات (طبيعية، تجانس التباينات).

3. اختيار الاختبار المناسب.

4. تنفيذ الاختبار.

5. تفسير النتائج في السياق البيولوجي.

Chapter Summary

خلاصة الفصل

Descriptive statistics are the first step in any data analysis. Measures of central tendency (mean, median, mode) and measures of dispersion (range, standard deviation, interquartile range) describe data in different ways. Choosing the right measure depends on the distribution shape and data type. Inspecting distributions using plots (histograms, boxplots, Q-Q plots) is an essential step before inference. Good data description protects against incorrect conclusions. Description and inference are linked: good description is the foundation of correct inference. The next chapter will move to probability and distributions as the foundation of statistical modeling.

الإحصاء الوصفي هو الخطوة الأولى في أي تحليل بيانات. مقاييس النزعة المركزية (المتوسط، الوسيط، المنوال) ومقاييس التشتت (المدى، الانحراف المعياري، المدى الربيعي) تصف البيانات بطرق مختلفة. اختيار المقياس المناسب يعتمد على شكل التوزيع ونوع البيانات. فحص التوزيعات باستخدام الرسوم (histograms، boxplots، Q-Q plots) هو خطوة أساسية قبل الاستدلال. الوصف الجيد للبيانات يحمي من الاستنتاجات الخاطئة. الوصف والاستدلال مرتبطان: الوصف الجيد هو أساس الاستدلال الصحيح. الفصل القادم سينتقل إلى الاحتمال والتوزيعات كأساس للنمذجة الإحصائية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Description Before Inference" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "الوصف قبل الاستدلال" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "descriptive statistics are sufficient to draw causal conclusions." This is incorrect; descriptive statistics summarize data, but causal conclusions require inference and experimental design. The correct statement is: "descriptive statistics summarize data and help understand its distribution, but causal conclusions require statistical tests and appropriate experimental design." A simple test: Practical check: "Does this descriptive measure tell me why, or does it only describe what is present?".

مثال على التجاوز: القول إن "الإحصاء الوصفي كافٍ لاستخلاص استنتاجات سببية". هذا غير صحيح؛ الوصفي يلخص البيانات، لكن الاستنتاجات السببية تتطلب استدلالاً وتصميماً تجريبياً. الصحيح هو: "الإحصاء الوصفي يلخص البيانات ويساعد في فهم توزيعها، لكن الاستنتاجات السببية تتطلب اختبارات إحصائية وتصميماً تجريبياً مناسباً."

اختبار بسيط: فحص تطبيقي: "هل هذا المقياس الوصفي يخبرني بالسبب، أم يصف فقط ما هو موجود؟"ـ

Example: Calculating the mean expression of a gene is a computational representation. Claiming that "this gene has an important biological role" is a biological claim not proven by the mean alone. The mean describes the central value, but it does not determine biological importance. Practical test: before inferring biological importance from a descriptive measure, Practical check: "Does this measure reflect a biologically meaningful difference, or is it just a numerical description?".

مثال: حساب متوسط التعبير لجين ما هو تمثيل حسابي. الادعاء بأن "هذا الجين له دور بيولوجي مهم" هو ادعاء بيولوجي لا يثبته المتوسط وحده. المتوسط يصف القيمة المركزية، لكنه لا يحدد الأهمية البيولوجية. اختبار عملي: قبل أن تستنتج أهمية بيولوجية من مقياس وصفي، فحص تطبيقي: "هل هذا المقياس يعكس اختلافاً ذا معنى بيولوجي، أم هو مجرد وصف رقمي؟"ـ

مثال تطبيقي على الإحصاء الوصفي:

· المدخلات: بيانات رقمية (مثل تعبير جيني)، نوع المتغير (مستمر، فئوي)ـ

· الافتراضات: أن البيانات تمثل العينة بدقة، وأن القيم المتطرفة مُحددة (إذا كانت موجودة)ـ

· المخرجات: مقاييس وصفية (متوسط، وسيط، انحراف معياري، IQR)، ورسوم بيانيةـ

· القيود: الإحصاء الوصفي لا يحدد الأهمية الإحصائية، ولا يفسر السببية، ويعتمد على جودة البياناتـ

Applied example of descriptive statistics:

اختبار: قبل الاعتماد على الإحصاء الوصفي، تحقق من جودة البيانات، وافحص التوزيع باستخدام الرسومـ

· Inputs: numerical data (e.g., gene expression), variable type (continuous, categorical).

· Assumptions: that the data accurately represents the sample, and that outliers are identified (if present).

· Outputs: descriptive measures (mean, median, standard deviation, IQR), and plots.

· Limitations: descriptive statistics do not determine statistical significance, nor explain causation, and depend on data quality.

Test: before relying on descriptive statistics, check data quality, and inspect the distribution using plots.

عند حساب الإحصاء الوصفي، يجب تسجيل:

· المصدر: البيانات المستخدمة (مصدرها، تاريخها، تحويلاتها).

· الإصدار: إصدار الأدوات المستخدمة (مثل Python، NumPy، pandas).

· المعلمات: أي معاملات (مثل العتبات، التحويلات).

· التحويلات: أي تحويلات طبقت على البيانات قبل الحساب (مثل تحويل لوغاريتمي).

When calculating descriptive statistics, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة حساب نفس المقاييس الوصفية؟"

إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the data used (its source, date, transformations).

· Version: the tool versions used (e.g., Python, NumPy, pandas).

· Parameters: any parameters (e.g., thresholds, transformations).

· Transformations: any transformations applied to the data before calculation (e.g., log transformation).

Test: Practical check: "If I gave this record to a colleague a year later, could they recalculate the same descriptive measures?" If the answer is no, the documentation is insufficient.

قبل الاعتماد على الإحصاء الوصفي في تحليل بيانات حقيقية، اختبره على بيانات اصطناعية ذات خصائص معروفة.

أنشئ بيانات تتبع توزيعات مختلفة (طبيعية، منحرفة، بقيم متطرفة) وتحقق من أن المقاييس الوصفية تعكس

الخصائص المتوقعة. إذا نجح الاختبار، فأنت واثق من أن حساباتك صحيحة، وأي نتيجة مع بيانات حقيقية تعكس

خصائص البيانات.

Before relying on descriptive statistics in real data analysis, test them on synthetic data with known properties. Create data following different distributions (normal, skewed, with outliers) and verify that the descriptive measures reflect the expected properties. If the test succeeds, the workflow is confident that the project’s calculations are correct, and any result with real data reflects the data's properties.

Chapter 49 — Probability and Distributions

الفصل 49 — الاحتمال والتوزيعات

الفكرة المركزية: probability, distributions, expectation, variance, modeling.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

A probability distribution is a model of uncertainty, not a copy of reality. Choosing a distribution imposes assumptions that can affect estimation and testing. Model adequacy should therefore be examined rather than treating the distribution name as a fact.

التوزيع الاحتمالي نموذج لعدم اليقين وليس نسخة من الواقع. اختيار توزيع معين يفرض افتراضات حول البيانات، ويمكن أن يؤثر في التقدير والاختبار. لذلك يجب فحص مدى ملاءمة النموذج وعدم التعامل مع اسم التوزيع كحقيقة بحد ذاته.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. What Is a Probability Distribution?

1. ما هو التوزيع الاحتمالي؟

A probability distribution is a mathematical function that describes the probability of obtaining different values of a random variable. In other words, it is a model of how data is distributed in a population. The distribution defines:

التوزيع الاحتمالي هو دالة رياضية تصف احتمالية الحصول على قيم مختلفة لمتغير عشوائي. بعبارة أخرى، هو نموذج لكيفية توزع البيانات في مجتمع ما. التوزيع يحدد:

· The possible values of the variable (what values can it take?).

· القيم الممكنة للمتغير (ما هي القيم التي يمكن أن يأخذها؟).

· The probabilities associated with each value (how likely is each value?).

· الاحتمالات المرتبطة بكل قيمة (ما مدى احتمالية كل قيمة؟).

مثال:

- متغير ثنائي (مثل نجاح/فشل): توزيع برنولي.

- عدد مرات النجاح في عدد ثابت من المحاولات: توزيع ثنائي الحدين (Binomial).

- عدد الأحداث في فترة زمنية: توزيع بواسون (Poisson).

- قياسات مستمرة: توزيع طبيعي (Normal).

Choosing the right distribution depends on the nature of the variable and the process that generates the data.

2. Why Are Probability Distributions Important in Biostatistics؟

اختيار التوزيع المناسب يعتمد على طبيعة المتغير والعملية التي تولد البيانات.

In biostatistics, probability distributions are the foundation of all statistical inference. They are used for:

2. لماذا التوزيعات الاحتمالية مهمة في الإحصاء الحيوي؟

في الإحصاء الحيوي، التوزيعات الاحتمالية هي أساس كل الاستدلال الإحصائي. تستخدم لـ:

· تقدير المعلمات: مثل تقدير متوسط تعبير جين معين باستخدام التوزيع الطبيعي.

· اختبار الفرضيات: مثل استخدام توزيع t لاختبار الفرق بين مجموعتين.

· Parameter estimation: such as estimating the mean expression of a specific gene using the normal distribution.

· بناء النماذج: مثل استخدام التوزيع السالب ذو الحدين (Negative Binomial) لنمذجة بيانات RNA-seq.

· Hypothesis testing: such as using the t-distribution to test the difference between two groups.

· حساب فترات الثقة (Confidence Intervals): لتقدير نطاق عدم اليقين حول تقدير معين.

· Model building: such as using the Negative Binomial distribution to model RNA-seq data.

· Calculating Confidence Intervals: to estimate the range of uncertainty around a specific estimate.

مبدأ: التوزيع الاحتمالي هو النموذج الذي يربط البيانات بعدم اليقين.

بدون نموذج، لا يمكننا قياس عدم اليقين.

3. Common Distributions in Biostatistics

There are several common distributions in biostatistics, each with its own properties and uses:

3. توزيعات شائعة في الإحصاء الحيوي

هناك عدة توزيعات شائعة في الإحصاء الحيوي، لكل منها خصائص واستخدامات:

The normal (or Gaussian) distribution is the most famous distribution. It has a bell shape, symmetric around the mean. It is used for modeling continuous measurements when data is symmetric. Many statistical tests (such as t-test, ANOVA) assume normal distribution or approximate it.

التوزيع الطبيعي (Normal Distribution)ـ

التوزيع الطبيعي (أو Gaussian) هو التوزيع الأكثر شهرة. له شكل الجرس، متماثل حول المتوسط. يُستخدم لنمذجة القياسات المستمرة عندما تكون البيانات متناظرة. العديد من الاختبارات الإحصائية (مثل t-test، ANOVA) تفترض التوزيع الطبيعي أو تقترب منه.

خصائص التوزيع الطبيعي:

· المتوسط = الوسيط = المنوال.

· 68% من البيانات تقع ضمن انحراف معياري واحد من المتوسط.

Properties:

· 95% ضمن انحرافين معياريين.

· Mean = Median = Mode.

· 99.7% ضمن ثلاثة انحرافات معيارية.

· 68% of data falls within one standard deviation of the mean.

· 95% within two standard deviations.

· 99.7% within three standard deviations.

```
#مثال: التوزيع الطبيعي
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
#توليد بيانات من توزيع طبيعي
mu, sigma = 10, 2
data = np.random.normal(mu, sigma, 1000)
# رسم التوزيع
plt.figure(figsize=(8, 4))
plt.hist(data, bins=30, density=True, alpha=0.7, label='بيانات')
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 100)
plt.plot(x, stats.norm.pdf(x, mu, sigma), 'r-', label='التوزيع الطبيعي')
plt.xlabel('القيمة')
plt.ylabel('الكثافة')
```

plt.title('التوزيع الطبيعي (المتوسط=10، الانحراف المعياري=2)')

```
plt.legend()
plt.show()
```

توزيع بواسون (Poisson Distribution).

The Poisson distribution is used for modeling the number of events occurring in a fixed time or space interval, when events are rare and independent. In bioinformatics, it is used for modeling read counts in RNA-seq data before normalization, because reads come from a limited number of molecules and approximately follow a Poisson distribution.

توزيع بواسون يستخدم لنمذجة عدد الأحداث التي تحدث في فترة زمنية أو مكانية محددة، عندما تكون الأحداث نادرة ومستقلة. في المعلوماتية الحيوية، يستخدم لنمذجة أعداد القراءات (read counts) في بيانات RNA-seq قبل التطبيع، لأن القراءات تأتي من عدد محدود من الجزيئات وتتبع توزيع بواسون تقريباً.

خصائص توزيع بواسون:

· المتوسط = التباين = λ (لامدا).

· مناسب للبيانات التي تمثل عدّ أحداث نادرة.

Properties:

· في RNA-seq، التباين غالباً أكبر من المتوسط (overdispersion)، مما يستدعي استخدام التوزيع السالب ذو الحدين.

· Mean = Variance = λ.

· Suitable for data representing counts of rare events.

· In RNA-seq, variance is often greater than the mean (overdispersion), requiring the Negative Binomial distribution.

```
# مثال: توزيع بواسون
lambda_poisson = 5
data_poisson = np.random.poisson(lambda_poisson, 1000)
plt.figure(figsize=(8, 4))
plt.hist(data_poisson, bins=range(0, 15), density=True, alpha=0.7, label='بيانات')
x = np.arange(0, 15)
plt.plot(x, stats.poisson.pmf(x, lambda_poisson), 'ro', label='توزيع بواسون (λ=5)')
plt.xlabel('العدد')
plt.ylabel('الاحتمال')
plt.title('توزيع بواسون')
plt.legend()
plt.show()
```

التوزيع السالب ذو الحدين (Negative Binomial Distribution).

The Negative Binomial distribution is a generalization of the Poisson distribution that allows variance greater than the mean (overdispersion). It is widely used in RNA-seq data analysis, especially in packages like DESeq2 and edgeR. This distribution assumes that counts follow a Poisson distribution with additional variance (biological and technical variation).

التوزيع السالب ذو الحدين هو تعميم لتوزيع بواسون يسمح بتباين أكبر من المتوسط (overdispersion). يستخدم على نطاق واسع في تحليل بيانات RNA-seq، خاصة في حزم مثل DESeq2 و edgeR. يفترض هذا التوزيع أن الأعداد تتبع توزيع بواسون مع تباين إضافي (تباين بيولوجي وتقني).

لماذا التوزيع السالب ذو الحدين في RNA-seq؟

- بيانات RNA-seq هي أعداد قراءات (counts).

- الأعداد تتبع توزيع بواسون تقريباً (متوسط = تباين).

- لكن في الواقع، التباين أكبر من المتوسط (بسبب التباين البيولوجي والتقني).

- التوزيع السالب ذو الحدين يسمح بتباين أكبر (معلمة التشتت).

4. Choosing the Right Model

Choosing the right distribution is not a random decision. It must be justified by the measurement process and experimental design. Some questions to ask:

4. اختيار النموذج المناسب

اختيار التوزيع المناسب ليس قراراً عشوائياً. يجب أن يكون مبرراً بعملية القياس والتصميم التجريبي. بعض الأسئلة التي يجب طرحها:

· ما هو نوع المتغير؟ (عدّ، قياس، فئوي).

· What type of variable is it? (count, measurement, categorical).

· كيف تولدت البيانات؟ (هل هي أحداث نادرة؟ هل هناك تباين إضافي؟).

· How were the data generated? (Are they rare events? Is there additional variation?).

· ما هي افتراضات الاختبارات التي سأستخدمها؟ (هل تفترض التوزيع الطبيعي؟).

· What are the assumptions of the tests I will use? (Do they assume normality?).

· هل البيانات تتبع التوزيع المختار؟ (استخدم الرسوم والاختبارات للتحقق).

· Does the data follow the chosen distribution? (Use plots and tests to verify).

قاعدة: لا تختار توزيعاً لمجرد أنه مألوف.

اختر التوزيع الذي يعكس عملية توليد البيانات بشكل أفضل.

5. Verifying Model Assumptions

After choosing a model, you must verify that the data conforms to its assumptions. Some tools for verification:

5. التحقق من افتراضات النموذج

بعد اختيار نموذج، يجب التحقق من أن البيانات تتوافق مع افتراضاته. بعض الأدوات للتحقق:

· الرسوم البيانية: مثل Q-Q plot للتحقق من التوزيع الطبيعي.

· Plots: such as Q-Q plot to check normality.

· الاختبارات الإحصائية: مثل اختبار Shapiro-Wilk للطبيعية.

· Statistical tests: such as Shapiro-Wilk test for normality.

· فحص التباين: مقارنة المتوسط والتباين للتحقق من افتراضات بواسون.

· Variance checking: comparing mean and variance to check Poisson assumptions.

· تحليل البقايا: في النماذج الانحدارية، فحص بقايا النموذج للتحقق من الافتراضات.

· Residual analysis: in regression models, checking model residuals to verify assumptions.

```
data = np.random.normal(10, 2, 100)
stats.probplot(data, dist="norm", plot=plt)
plt.title('Q-Q Plot للتحقق من الطبيعية')
plt.show()
# مثال: التحقق من التوزيع الطبيعي باستخدام Q-Q plot
from scipy import stats
If the points are close to the straight line, the data is approximately normal. If they are skewed, you may need a transformation or a different model.
```

6. Practical Application: Choosing a Distribution for RNA-seq Data

إذا كانت النقاط قريبة من الخط المستقيم، فالبيانات طبيعية تقريباً. إذا كانت منحرفة، قد تحتاج إلى تحويل أو

استخدام نموذج آخر.

Let us apply these principles to simulated RNA-seq data:

6. التطبيق العملي: اختيار توزيع لبيانات RNA-seq

لنطبق هذه المبادئ على بيانات RNA-seq محاكاة:

```
# محاكاة بيانات RNA-seq (أعداد قراءات)
np.random.seed(42)
```

n_genes = 1000

n_samples = 20

توليد أعداد من توزيع بواسون

counts_poisson = np.random.poisson(lam=50, size=(n_genes, n_samples))

توليد أعداد مع تباين زائد (محاكاة Negative Binomial)

counts_nb = np.random.negative_binomial(n=10, p=0.2, size=(n_genes, n_samples))

حساب المتوسط والتباين

```
print("بيانات بواسون:")
print(f"  متوسط التباين: {np.mean(np.var(counts_poisson, axis=1)):.2f}")
print(f"  متوسط المتوسط: {np.mean(np.mean(counts_poisson, axis=1)):.2f}")
print(f"  نسبة التباين/المتوسط: {np.mean(np.var(counts_poisson, axis=1) / np.mean(counts_poisson, axis=1)):.2f}")
print("\nبيانات Negative Binomial (تباين زائد):")
print(f"  متوسط التباين: {np.mean(np.var(counts_nb, axis=1)):.2f}")
print(f"  متوسط المتوسط: {np.mean(np.mean(counts_nb, axis=1)):.2f}")
print(f"  نسبة التباين/المتوسط: {np.mean(np.var(counts_nb, axis=1) / np.mean(counts_nb, axis=1)):.2f}")
# تفسير:
# في بيانات بواسون، التباين ≈ المتوسط (النسبة ≈ 1)
# في بيانات Negative Binomial، التباين > المتوسط (النسبة > 1)
```

This example shows why the Negative Binomial distribution is suitable for RNA-seq data: variance is often greater than the mean.

هذا المثال يوضح لماذا التوزيع السالب ذو الحدين مناسب لبيانات RNA-seq: التباين غالباً أكبر من المتوسط.

7. Limits of Probability Models

Probability models are powerful tools, but they have limits:

7. حدود النماذج الاحتمالية

النماذج الاحتمالية أدوات قوية، لكن لها حدود:

· تعتمد على افتراضات: النموذج الجيد يعتمد على افتراضات قد لا تنطبق دائماً.

· They depend on assumptions: a good model relies on assumptions that may not always hold.

· قد تكون مفرطة في التبسيط: النماذج تبسط الواقع ليكون قابلاً للتحليل، وهذا قد يفقد تفاصيل مهمة.

· They may be oversimplified: models simplify reality to be analyzable, which may lose important details.

· تعتمد على جودة البيانات: النموذج لا يصحح البيانات السيئة.

· They depend on data quality: the model does not correct bad data.

· التفسير يعتمد على السياق: القيم الاحتمالية تحتاج إلى تفسير في السياق البيولوجي.

· Interpretation depends on context: probability values need interpretation in the biological context.

ملخص: النموذج الاحتمالي هو أداة لقياس عدم اليقين، وليس حقيقة مطلقة.

استخدمه بوعي، وافحص افتراضاته، وفسر نتائجه في السياق.

Chapter Summary

خلاصة الفصل

Probability distributions are models for uncertainty. The normal distribution fits symmetric continuous measurements. The Poisson distribution fits counts of rare events. The Negative Binomial distribution fits RNA-seq data where variance is greater than the mean. Model choice should be justified by the measurement process and experimental design. Verifying model assumptions is an essential step. Probability models are powerful tools, but they have limits, and depend on data quality and the validity of assumptions. The next chapter will move to Hypothesis Testing as a practical application of probability distributions.

التوزيعات الاحتمالية هي نماذج لعدم اليقين. التوزيع الطبيعي يناسب القياسات المستمرة المتناظرة. توزيع بواسون يناسب عد الأحداث النادرة. التوزيع السالب ذو الحدين يناسب بيانات RNA-seq حيث التباين أكبر من المتوسط. اختيار النموذج يجب أن يكون مبرراً بعملية القياس والتصميم التجريبي. التحقق من افتراضات النموذج خطوة أساسية. النماذج الاحتمالية أدوات قوية، لكن لها حدود، وتعتمد على جودة البيانات وصحة الافتراضات. الفصل القادم سينتقل إلى اختبار الفرضيات كتطبيق عملي للتوزيعات الاحتمالية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Probability and Distributions" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "الاحتمال والتوزيعات" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "the normal distribution fits all types of data." This is incorrect; the normal distribution is only suitable for symmetric continuous data. The correct statement is: "the normal distribution is suitable for symmetric continuous data, while other data types require different distributions (such as Poisson for counts, and Negative Binomial for RNA-seq data)." A simple test: Practical check: "Is my data continuous and symmetric, or is it count data or skewed?".

مثال على التجاوز: القول إن "التوزيع الطبيعي يناسب جميع أنواع البيانات". هذا غير صحيح؛ التوزيع الطبيعي مناسب للبيانات المتناظرة المستمرة فقط. الصحيح هو: "التوزيع الطبيعي مناسب للبيانات المستمرة المتناظرة، بينما تتطلب أنواع البيانات الأخرى توزيعات مختلفة (مثل بواسون للعدّ، والسالب ذو الحدين لبيانات RNA-seq)." اختبار بسيط: فحص تطبيقي: "هل بياناتي مستمرة ومتناظرة، أم أنها عدّ أو منحرفة؟"ـ

Example: Assuming that expression data follows a Negative Binomial distribution is a computational representation of the model. Claiming that "this model proves a biological difference" is a biological claim not proven by the model alone. The model defines how to measure uncertainty, but proving biological difference requires contextual interpretation. Practical test: before inferring a biological difference from a statistical model, Practical check: "Is this difference large enough to be biologically meaningful?".

مثال: افتراض أن بيانات التعبير تتبع توزيعاً سلبياً ذا حدين هو تمثيل حسابي للنموذج. الادعاء بأن "هذا النموذج يثبت وجود اختلاف بيولوجي" هو ادعاء بيولوجي لا يثبته النموذج وحده. النموذج يحدد كيفية قياس عدم اليقين، لكن إثبات الاختلاف البيولوجي يتطلب تفسيراً في السياق. اختبار عملي: قبل أن تستنتج وجود اختلاف بيولوجي من نموذج إحصائي، فحص تطبيقي: "هل هذا الاختلاف كبير بما يكفي ليكون ذا معنى بيولوجي؟"ـ

مثال تطبيقي على اختيار توزيع لبيانات RNA-seq:

· المدخلات: أعداد قراءات (counts) لكل جين في كل عينة.

· الافتراضات: أن الأعداد تتبع توزيعاً سلبياً ذا حدين (تباين > متوسط)، وأن التطبيع قد تم بشكل مناسب.

· المخرجات: نموذج إحصائي يصف توزيع البيانات، مع تقديرات للمعلمات (متوسط، تباين، معلمة التشتت).

· القيود: النموذج يعتمد على افتراضات قد لا تنطبق على جميع الجينات (مثل الجينات منخفضة التعبير)، وقد يحتاج إلى تعديلات.

Applied example of choosing a distribution for RNA-seq data:

اختبار: قبل اختيار نموذج، تحقق من توزيع الأعداد الخام، واستخدم رسوم تشخيصية (مثل متوسط-تباين plot) لتقييم مدى ملاءمة النموذج.

· Inputs: read counts for each gene in each sample.

· Assumptions: that counts follow a Negative Binomial distribution (variance > mean), and that normalization has been properly performed.

· Outputs: a statistical model describing the data distribution, with parameter estimates (mean, variance, dispersion parameter).

· Limitations: the model depends on assumptions that may not apply to all genes (e.g., low-expression genes), and may require adjustments.

Test: before choosing a model, check the distribution of raw counts, and use diagnostic plots (such as mean-variance plots) to assess model fit.

عند اختيار نموذج احتمالي، يجب تسجيل:

· المصدر: البيانات المستخدمة، وتحويلاتها.

· الإصدار: إصدار الأدوات المستخدمة (مثل R، DESeq2، scipy).

· المعلمات: معلمات النموذج (مثل المتوسط، التباين، معلمة التشتت).

· التحويلات: أي تحويلات طبقت على البيانات قبل النمذجة (مثل تحويل لوغاريتمي).

When choosing a probability model, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة بناء نفس النموذج والحصول على نفس التقديرات؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the data used, and its transformations.

· Version: the tool versions used (e.g., R, DESeq2, scipy).

· Parameters: model parameters (e.g., mean, variance, dispersion parameter).

· Transformations: any transformations applied to the data before modeling (e.g., log transformation).

Test: Practical check: "If I gave this record to a colleague a year later, could they rebuild the same model and obtain the same estimates?" If the answer is no, the documentation is insufficient.

قبل اعتماد نموذج احتمالي على بيانات حقيقية، اختبر النموذج على بيانات اصطناعية ذات خصائص معروفة. أنشئ بيانات من توزيعات مختلفة، وطبق نماذج مختلفة، وتحقق من أن النموذج المناسب يكتشف الخصائص الصحيحة.

هذا يساعدك على فهم تأثير اختيار النموذج على النتائج، ويحميك من استخدام نموذج غير مناسب.

Before relying on a probability model on real data, test the model on synthetic data with known properties. Create data from different distributions, apply different models, and verify that the appropriate model detects the correct properties. This helps you understand the impact of model choice on results, and protects you from using an inappropriate model.

Chapter 50 — Hypothesis Testing

الفصل 50 — اختبار الفرضيات

الفكرة المركزية: null، alternative، p-value، power، uncertainty.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

اختبار الفرضية لا يثبت الفرضية البديلة بصورة مطلقة. هو يقارن ما لوحظ بما يتوقعه نموذج تحت فرضية صفرية، وفق افتراضات محددة. معنى النتيجة يعتمد على تصميم الدراسة، وحجم الأثر، وعدم اليقين، والمقارنات الأخرى، وليس على قيمة p وحدها.

A hypothesis test does not prove an alternative hypothesis absolutely. It compares observations with what a model predicts under a null hypothesis and specified assumptions. Its meaning depends on study design, effect size, uncertainty, and other comparisons, not on a p-value alone.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. What Is Hypothesis Testing؟

1. ما هو اختبار الفرضيات؟

Hypothesis testing is a statistical procedure that makes a decision about a hypothesis based on sample data. It involves the following steps:

اختبار الفرضيات هو إجراء إحصائي يتخذ قراراً بشأن فرضية بناءً على بيانات العينة. يتضمن الخطوات التالية:

الخطوة الأولى: صياغة الفرضيات

· الفرضية العدمية (H_0): تفترض عدم وجود تأثير أو اختلاف. مثال: "لا يوجد فرق في التعبير بين المجموعتين".

· الفرضية البديلة (H_1): تفترض وجود تأثير أو اختلاف. مثال: "يوجد فرق في التعبير بين المجموعتين".

الخطوة الثانية: اختيار اختبار إحصائي.

يعتمد على نوع البيانات وتوزيعها وتصميم التجربة. أمثلة: اختبار t، ANOVA، اختبار مربع كاي.

الخطوة الثالثة: حساب قيمة الاختبار وقيمة p.

قيمة p هي احتمال الحصول على نتيجة متطرفة أو أكثر تطرفاً مما لدينا، بافتراض أن الفرضية العدمية صحيحة.

Step One: Formulating Hypotheses.

الخطوة الرابعة: اتخاذ القرار

· Null Hypothesis (H_0): assumes no effect or difference. Example: "There is no difference in expression between the two groups.".

مقارنة قيمة p مع مستوى دلالة محدد مسبقاً (α، عادة 0.05). إذا كانت $p < \alpha$، نرفض الفرضية العدمية. إذا كانت p

$\geq \alpha$، لا نرفض الفرضية العدمية.

· Alternative Hypothesis (H_1): assumes an effect or difference. Example: "There is a difference in expression between the two groups.".

Step Two: Choosing a Statistical Test.

Depends on data type, distribution, and experimental design. Examples: t-test, ANOVA, chi-square test.

Step Three: Calculating the Test Statistic and p-value.

The p-value is the probability of obtaining a result as extreme or more extreme than what we have, assuming the null hypothesis is true.

Step Four: Making a Decision

Compare the p-value to a pre-specified significance level (α, usually 0.05). If $p < \alpha$, reject the null hypothesis. If $p \geq \alpha$, do not reject the null hypothesis.

Language / نوع الكود: **Text / Hypothesis Example**

مثال مبسط:

H_0: متوسط التعبير في المجموعة A = متوسط التعبير في المجموعة B.

H_1: متوسط التعبير في المجموعة A ≠ متوسط التعبير في المجموعة B.

إذا كانت $p < 0.05$، نرفض H_0 ونستنتج أن هناك فرقاً ذا دلالة إحصائية.

2. The p-value: What It Means and What It Does Not Mean

The p-value is one of the most misunderstood concepts in statistics. Let us clarify what it means and what it does not mean:

2. قيمة p (p-value): ماذا تعني وماذا لا تعني؟

قيمة p هي أحد أكثر المفاهيم إساءة فهم في الإحصاء. دعنا نوضح ما تعنيه وما لا تعنيه:

ماذا تعني قيمة p؟

· قيمة p هي احتمال الحصول على بيانات متطرفة أو أكثر تطرفاً مما لدينا، بافتراض أن الفرضية العدمية صحيحة.

What does the p-value mean?

· هي مقياس لمدى توافق البيانات مع الفرضية العدمية.

· The p-value is the probability of obtaining data as extreme or more extreme than what we have, assuming the null hypothesis is true.

· قيمة p صغيرة تشير إلى أن البيانات غير متوافقة مع الفرضية العدمية.

· It is a measure of how compatible the data is with the null hypothesis.

· A small p-value indicates that the data is incompatible with the null hypothesis.

ماذا لا تعني قيمة p؟

· ليست احتمال أن الفرضية العدمية صحيحة. هذا هو الاحتمال البعدي (posterior probability) الذي يتطلب نظرية بايز.

· ليست احتمال أن الفرضية البديلة صحيحة.

What does the p-value NOT mean?

· ليست مقياساً لحجم الأثر أو الأهمية البيولوجية.

· It is not the probability that the null hypothesis is true. This is the posterior probability, which requires Bayesian inference.

· ليست مقياساً لتكرارية النتيجة (replicability).

· It is not the probability that the alternative hypothesis is true.

· It is not a measure of effect size or biological significance.

· It is not a measure of replicability.

Language / نوع الكود: **Text / Statistical Interpretation**

مثال توضيحي:

قيمة $p = 0.03$ تعني: إذا كانت الفرضية العدمية صحيحة، فإن احتمال رؤية بيانات بهذا التطرف (أو أكثر) هو 3%.

لا تعني: هناك احتمال 3% أن الفرضية العدمية صحيحة.

لا تعني: هناك احتمال 97% أن الفرضية البديلة صحيحة.

```
# محاكاة: فهم قيمة p
import numpy as np
from scipy import stats
# توليد بيانات من مجموعتين (لا فرق حقيقي)
```

```
np.random.seed(42)
group1 = np.random.normal(10, 2, 30)
group2 = np.random.normal(10, 2, 30)
# اختبار t
t_stat, p_value = stats.ttest_ind(group1, group2)
print(f"قيمة p: {p_value:.4f}")
# قيمة p عندما لا يوجد فرق حقيقي (<< 0.05) عادة ما تكون كبيرة
```

3. Confidence Intervals

A confidence interval is a range of values that estimates a population parameter (such as the mean) with a certain level of confidence (usually 95%). A 95% confidence interval means that we expect 95% of intervals constructed this way to contain the true parameter. A confidence interval provides more information than a p-value: it tells us about the effect size and its precision.

3. فواصل الثقة (Confidence Intervals)

فاصل الثقة هو نطاق من القيم يُقدّر معلمة مجتمعية (مثل المتوسط) بدرجة معينة من الثقة (عادة 95%). فاصل الثقة 95% يعني أننا نتوقع أن 95% من الفواصل المبنية بهذه الطريقة ستحتوي على المعلمة الحقيقية. فاصل الثقة يعطي معلومات أكثر من قيمة p: يخبرنا بحجم التأثير ودقته.

مثال:

تقدير متوسط الفرق بين مجموعتين = 2.5، فاصل ثقة 95%: [1.2, 3.8].

هذا يعني أننا واثقون بنسبة 95% أن الفرق الحقيقي يقع بين 1.2 و 3.8.

إذا كان الفاصل لا يحتوي على صفر، فهذا يتوافق مع دلالة إحصائية.

إذا كان يحتوي على صفر، فلا يمكن استبعاد عدم وجود فرق.

```
# بيانات محاكاة
group1 = np.random.normal(10, 2, 30)
group2 = np.random.normal(12, 2, 30)
# حساب فاصل ثقة للفرق بين متوسطين
from scipy import stats
# وحساب فاصل الثقة t اختبار
t_stat, p_value = stats.ttest_ind(group1, group2)
df = len(group1) + len(group2) - 2
mean_diff = np.mean(group2) - np.mean(group1)
se = np.sqrt(np.var(group1)/len(group1) + np.var(group2)/len(group2))
ci_lower = mean_diff - stats.t.ppf(0.975, df) * se
ci_upper = mean_diff + stats.t.ppf(0.975, df) * se
print(f"متوسط الفرق: {mean_diff:.2f}")
print(f"95 فاصل الثقة %: [{ci_lower:.2f}, {ci_upper:.2f}]")
```

4. Effect Size

Effect size is a measure of the magnitude of the difference or effect, independent of sample size. While the p-value tells us whether a difference exists, effect size tells us how large that difference is. Common effect size measures include:

4. حجم الأثر (Effect Size)

Cohen's d: the difference between two means divided by the pooled standard deviation. Values of 0.2 = small, 0.5 = medium, 0.8 = large.

حجم الأثر هو مقياس لمقدار الاختلاف أو التأثير، مستقل عن حجم العينة. بينما تخبرنا قيمة p بوجود اختلاف، يخبرنا حجم الأثر بمدى هذا الاختلاف. مقاييس حجم الأثر الشائعة تشمل:

R^2: the proportion of variance explained in regression models.

Cohen's d: الفرق بين متوسطين مقسوماً على الانحراف المعياري المشترك. قيم 0.2 = صغير، 0.5 = متوسط، 0.8 = كبير.

Odds Ratio: common in clinical and epidemiological studies.

R^2: نسبة التباين المفسر في النماذج الانحدارية.

Odds Ratio (نسبة الأرجحية): شائعة في الدراسات السريرية والوبائية.

```
# حساب Cohen's d
def cohens_d(group1, group2):
    n1, n2 = len(group1), len(group2)
    var1, var2 = np.var(group1, ddof=1), np.var(group2, ddof=1)
    pooled_var = ((n1 - 1) * var1 + (n2 - 1) * var2) / (n1 + n2 - 2)
    pooled_std = np.sqrt(pooled_var)
    return abs(np.mean(group1) - np.mean(group2)) / pooled_std
d = cohens_d(group1, group2)
print(f"Cohen's d: {d:.2f}")
```

الدوال المعرفة هنا (cohens_d) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام.

5. Statistical Power

Statistical power is the probability of rejecting the null hypothesis when it is false (i.e., detecting a true effect). Power depends on:

5. القوة الإحصائية (Statistical Power)

القوة الإحصائية هي احتمال رفض الفرضية العدمية عندما تكون خاطئة (أي اكتشاف تأثير حقيقي). تعتمد القوة على:

· حجم الأثر: الأثر الأكبر يسهل اكتشافه.

· Effect size: larger effects are easier to detect.

· حجم العينة: العينات الأكبر تزيد القوة.

· Sample size: larger samples increase power.

· مستوى الدلالة (α): α الأصغر يقلل القوة.

· Significance level (α): smaller α decreases power.

· التباين: التباين الأكبر يقلل القوة.

· Variance: larger variance decreases power.

مثال:

إذا كانت القوة 0.8، فهذا يعني أن هناك احتمال 80% لاكتشاف تأثير حقيقي بحجم معين.

القوة المنخفضة تعني خطراً أكبر لتفويت تأثير حقيقي (خطأ من النوع الثاني).

6. Interpreting Results: Interpretation Instead of Threshold Worship

A common mistake is treating the p-value as a threshold (such as 0.05) that separates "significant" from "non-significant." This oversimplification loses a lot of information. Good interpretation depends on:

6. تفسير النتائج: التفسير بدلاً من عبادة العتبات

من الأخطاء الشائعة التعامل مع قيمة p كحد فاصل (مثل 0.05) يفصل بين "مهم" و"غير مهم". هذا الاختزال يخسر الكثير من المعلومات. التفسير الجيد يعتمد على:

· حجم الأثر: ما هو حجم الاختلاف؟ هل هو ذو معنى بيولوجي؟

· فاصل الثقة: ما هو نطاق عدم اليقين؟

· Effect size: how large is the difference? Is it biologically meaningful?

· التصميم التجريبي: هل العينات ممثلة؟ هل هناك مكررات كافية؟

· Confidence interval: what is the range of uncertainty?

· السياق البيولوجي: هل النتيجة متسقة مع المعرفة الموجودة؟

· Experimental design: are samples representative? Are there sufficient replicates?

· قابلية التكرار: هل يمكن تكرار النتيجة في دراسة مستقلة؟

· Biological context: is the result consistent with existing knowledge?

· Replicability: can the result be repeated in an independent study?

قاعدة: قيمة p أداة، وليس حكماً نهائياً.

استخدمها مع حجم الأثر وفاصل الثقة والسياق لاتخاذ قرار مستنير.

7. Practical Application: Hypothesis Testing in Expression Data

Let us apply hypothesis testing to simulated expression data:

7. التطبيق العملي: اختبار فرضية في بيانات تعبير

لنطبق اختبار الفرضيات على بيانات تعبير محاكاة:

```
# محاكاة بيانات تعبير: 100 جين، 20 عينة (10 سليمة، 10 مريضة)
np.random.seed(42)
n_genes = 100
n_samples_per_group = 10
# اختبار فرضية لكل جين
results = []
for i in range(n_genes):
    group1 = expression_data[i, :10]
    group2 = expression_data[i, 10:]
    t_stat, p_val = stats.ttest_ind(group1, group2)
    mean_diff = np.mean(group2) - np.mean(group1)
    d = cohens_d(group1, group2)
    results.append({
        "gene": i,
        "p_value": p_val,
        "mean_diff": mean_diff,
        "cohens_d": d
    })
# توليد البيانات (بعض الجينات مختلفة بين المجموعتين)
expression_data = np.random.normal(10, 2, (n_genes, 20))
for i in range(20):  # 20 جين مختلف
    expression_data[i, 10:] += 3  # زيادة في المجموعة المريضة
# تصحيح للاختبارات المتعددة (محاكاة Benjamini-Hochberg)
from statsmodels.stats.multitest import multipletests
p_values = [res["p_value"] for res in results]
rejected, p_adj, _, _ = multipletests(p_values, alpha=0.05, method='fdr_bh')
# عرض أول 5 نتائج
print("أول 5 جينات - نتائج اختبار الفرضيات:")
for res in results[:5]:
    print(f"جين {res['gene']}: p={res['p_value']:.4f}, "
          f"الفرق={res['mean_diff']:.2f}, d={res['cohens_d']:.2f}")
```

```
# عدد الجينات المكتشفة
sig_count = sum(rejected)
print(f"\nعدد الجينات المكتشفة كذات تعبير تفاضلي: {sig_count}")
```

This example shows how to apply hypothesis testing, calculate effect size, and correct for multiple testing.

8. Limits of Hypothesis Testing

هذا المثال يوضح كيفية تطبيق اختبار الفرضيات، وحساب حجم الأثر، وتصحيح الاختبارات المتعددة.

Hypothesis testing is a powerful tool, but it has limits:

8. حدود اختبار الفرضيات

اختبار الفرضيات أداة قوية، لكن له حدود:

· يعتمد على الافتراضات: إذا انتهكت الافتراضات، فقد تكون النتائج غير موثوقة.

· يعتمد على حجم العينة: العينات الكبيرة قد تعطي قيماً p صغيرة جداً لتأثيرات صغيرة غير ذات معنى بيولوجي.

· Depends on assumptions: if assumptions are violated, results may be unreliable.

· لا يخبرك بالسبب: يخبرك بوجود اختلاف، لكنه لا يشرح لماذا.

· Depends on sample size: large samples may yield very small p-values for tiny effects of no biological significance.

· قد يُساء تفسيره: قيمة p ليست احتمال صحة الفرضية.

· Does not tell you why: it tells you a difference exists, but does not explain why.

· لا يعوض عن تصميم تجربة سيء: لا يمكن للاختبار الإحصائي إصلاح عيوب التصميم.

· Can be misinterpreted: the p-value is not the probability of the hypothesis being true.

· Does not compensate for poor experimental design: statistical tests cannot fix design flaws.

ملخص: استخدم اختبار الفرضيات كأداة، وليس كحكم نهائي.

اجمعه مع حجم الأثر، وفاصل الثقة، والسياق البيولوجي.

Chapter Summary

خلاصة الفصل

Hypothesis testing is a statistical procedure for evaluating hypotheses using p-values, confidence intervals, effect size, and statistical power. The p-value is the

probability of obtaining extreme data assuming the null hypothesis is true, not the probability of the hypothesis being true. Confidence intervals give a range for uncertainty. Effect size measures the magnitude of the difference. Statistical power measures the ability to detect a true effect. Good interpretation goes beyond the 0.05 threshold to include effect size, confidence intervals, biological context, and replicability. Hypothesis testing is a powerful tool but has limits, and depends on data quality and the validity of assumptions. The next chapter will move to multiple comparisons, an extension of hypothesis testing in the context of genomic data.

اختبار الفرضيات هو إجراء إحصائي لتقييم الفرضيات باستخدام قيمة p وفواصل الثقة وحجم الأثر والقوة الإحصائية. قيمة p هي احتمال الحصول على بيانات متطرفة بافتراض صحة الفرضية العدمية، وليست احتمال صحة الفرضية. فاصل الثقة يعطي نطاقاً لعدم اليقين. حجم الأثر يقيس مقدار الاختلاف. القوة الإحصائية تقيس القدرة على اكتشاف تأثير حقيقي. التفسير الجيد يتجاوز عتبة 0.05 ليشمل حجم الأثر، وفاصل الثقة، والسياق البيولوجي، وقابلية التكرار. اختبار الفرضيات أداة قوية لكن لها حدود، وتعتمد على جودة البيانات وصحة الافتراضات. الفصل القادم سينتقل إلى المقارنات المتعددة، وهي امتداد لاختبار الفرضيات في سياق البيانات الجينومية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Hypothesis Testing" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "اختبار الفرضيات" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "a small p-value proves that the alternative hypothesis is true." This is incorrect; the p-value measures compatibility with the null hypothesis, not the probability of the alternative hypothesis being true. The correct statement is: "a small p-value indicates that the data is incompatible with the null hypothesis, providing evidence against it, but proving the alternative hypothesis requires contextual interpretation and additional evidence." A simple test: Practical check: "Does the p-value tell me the probability that my hypothesis is true, or does it only tell me how compatible my data is with the null hypothesis?".

مثال على التجاوز: القول إن "قيمة p الصغيرة تثبت أن الفرضية البديلة صحيحة". هذا غير صحيح؛ قيمة p تقيس التوافق مع الفرضية العدمية، وليس احتمال صحة الفرضية البديلة. الصحيح هو: "قيمة p الصغيرة تشير إلى أن البيانات غير متوافقة مع الفرضية العدمية، مما يوفر دليلاً ضدها، لكن إثبات صحة الفرضية البديلة يتطلب تفسيراً في

السياق وأدلة إضافية." اختبار بسيط: فحص تطبيقي: "هل قيمة p تخبرني باحتمال أن تكون فرضيتي صحيحة، أم تخبرني فقط بمدى توافق بياناتي مع الفرضية العدمية؟"ـ

Example: A t-test result showing p = 0.01 is a computational representation. Claiming that "this gene causes the disease" is a biological claim not proven by the statistical test alone. The test shows a difference, but proving a causal role requires additional evidence. Practical test: before inferring a biological role from a statistical result, Practical check: "Is there independent evidence supporting this role?".

مثال: نتيجة اختبار t تعطي p = 0.01 هي تمثيل حسابي. الادعاء بأن "هذا الجين يسبب المرض" هو ادعاء بيولوجي لا يثبته الاختبار الإحصائي وحده. الاختبار يظهر اختلافاً، لكن إثبات الدور السببي يحتاج إلى أدلة إضافية. اختبار عملي: قبل أن تستنتج دوراً بيولوجياً من نتيجة إحصائية، فحص تطبيقي: "هل هناك أدلة مستقلة تدعم هذا الدور؟"ـ

مثال تطبيقي على اختبار t:

· المدخلات: بيانات عددية من مجموعتين (مستقلة)، مستوى دلالة (α)ـ

· الافتراضات: أن البيانات تتبع توزيعاً طبيعياً تقريباً، وأن التباينات متساوية تقريباً، وأن العينات مستقلةـ

· المخرجات: قيمة t، قيمة p، وفاصل ثقة للفرق (إذا طُلب)ـ

· القيود: الاختبار حساس للقيم المتطرفة، وقد لا يكون مناسباً للبيانات المنحرفة أو ذات التباين غير المتساويـ

Applied example of a t-test:

اختبار: قبل تطبيق اختبار t، تحقق من التوزيع باستخدام الرسوم (Q-Q plot، boxplot) ومن تجانس التباينات باستخدام اختبار Leveneـ

· Inputs: numerical data from two groups (independent), significance level (α).

· Assumptions: that the data approximately follows a normal distribution, that variances are approximately equal, and that samples are independent.

· Outputs: t-statistic, p-value, and confidence interval for the difference (if requested).

· Limitations: the test is sensitive to outliers, and may not be suitable for skewed data or data with unequal variances.

Test: before applying a t-test, check the distribution using plots (Q-Q plot, boxplot) and check homogeneity of variances using Levene's test.

عند إجراء اختبار الفرضيات، يجب تسجيل:

· المصدر: البيانات المستخدمة، وتحويلاتها.

· الإصدار: إصدار الأدوات المستخدمة (مثل scipy، statsmodels، R).

· المعلمات: مستوى الدلالة (α)، نوع الاختبار (أحادي أو ثنائي الطرف).

· التحويلات: أي تحويلات طبقت على البيانات قبل الاختبار.

When conducting hypothesis testing, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة نفس الاختبار والحصول على نفس النتائج؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Version: the tool versions used (e.g., scipy, statsmodels, R).

· Parameters: the significance level (α), the type of test (one-tailed or two-tailed).

· Transformations: any transformations applied to the data before the test.

Test: Practical check: "If I gave this record to a colleague a year later, could they rerun the same test and obtain the same results?" If the answer is no, the documentation is insufficient.

قبل تطبيق اختبار الفرضيات على بيانات حقيقية، اختبره على بيانات اصطناعية ذات خصائص معروفة. أنشئ بيانات بدون فرق حقيقي (تحت H_0) وبيانات بفرق معروف (تحت H_1)، وتأكد من أن الاختبار يتصرف كما هو متوقع (قيمة p موزعة بشكل منتظم تحت H_0، وقوة معقولة تحت H_1). هذا يساعدك على فهم سلوك الاختبار والتحقق من صحته.

Before applying hypothesis testing to real data, test it on synthetic data with known properties. Create data with no real difference (under H_0) and data with a known difference (under H_1), and verify that the test behaves as expected (p-values uniformly distributed under H_0, reasonable power under H_1). This helps you understand the test's behavior and verify its validity.

Chapter 51 — Multiple Comparisons

الفصل 51 — المقارنات المتعددة

Deeper Conceptual Explanation

الفكرة المركزية: family-wise error، FDR، correction، genomic scale.

شرح أعمق للمفهوم

When many hypotheses are tested, the chance of seeing apparently important results by chance increases. Multiple-comparison methods alter decision criteria

or control error rates according to the goal. No single method is always best; the choice should follow the question, design, and costs of different errors.

عندما تختبر فرضيات كثيرة، ترتفع فرصة ظهور نتيجة تبدو مهمة بالصدفة. طرق التصحيح تغير معيار القرار أو تضبط معدل الأخطاء وفق الهدف. لا توجد طريقة واحدة هي الأفضل دائماً؛ الاختيار يجب أن يتبع السؤال وتصميم الدراسة وكلفة الأخطاء من النوع الأول والثاني.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. The Multiple Comparisons Problem

When testing a single hypothesis at α = 0.05, the Type I error probability is 5%. That is, there is a 5% chance of rejecting a true null hypothesis (false positive). But when we test 10,000 hypotheses (such as 10,000 genes), the situation changes.

1. مشكلة المقارنات المتعددة

عند اختبار فرضية واحدة بمستوى دلالة α = 0.05، يكون احتمال الخطأ من النوع الأول 5%. أي أن هناك احتمال 5% لرفض فرضية عدمية صحيحة (إيجابية كاذبة). لكن عندما نختبر 10,000 فرضية (مثل 10,000 جين)، يصبح الوضع مختلفاً.

إذا كانت الفرضيات العدمية صحيحة لجميع الجينات (لا يوجد فرق حقيقي):

- لكل جين، احتمال إيجابية كاذبة = 0.05.

- عدد الإيجابيات الكاذبة المتوقع = 10,000 × 0.05 = 500.

- أي أننا نتوقع حوالي 500 جيناً يظهر كذات تعبير تفاضلي، على الرغم من عدم وجود أي فرق حقيقي.

This is the essence of the problem. Without correction, we would have many false positives that could lead to incorrect biological conclusions.

هذا هو جوهر المشكلة. بدون تصحيح، سيكون لدينا العديد من الإيجابيات الكاذبة التي قد تؤدي إلى استنتاجات بيولوجية خاطئة.

```
# محاكاة: الإيجابيات الكاذبة المتوقعة
n_tests = 10000
alpha = 0.05
expected_false_positives = n_tests * alpha
print(f"الإيجابيات الكاذبة المتوقعة: {expected_false_positives}")
# الناتج: 500
```

2. Family-Wise Error Rate (FWER)

The Family-Wise Error Rate is the probability of making at least one Type I error among all the tests we perform. The classic method for controlling FWER is the Bonferroni correction.

2. التحكم في معدل الخطأ العائلي (FWER)

Bonferroni method: multiply each p-value by the number of tests (or divide the significance level by the number of tests).

معدل الخطأ العائلي هو احتمال ارتكاب خطأ من النوع الأول واحد على الأقل بين جميع الاختبارات التي نجريها. الطريقة الكلاسيكية للتحكم في FWER هي تصحيح بونفيروني (Bonferroni correction).

طريقة بونفيروني: يتم ضرب قيمة p لكل اختبار في عدد الاختبارات (أو تقسيم مستوى الدلالة على عدد الاختبارات).

مثال:

- عدد الاختبارات = 10,000.

- المعدل α = 0.05 / 10,000 = 0.000005.

- أي أن الجين لا يُعتبر مختلفاً إلا إذا كانت $p < 0.000005$.

مزايا بونفيروني:

· بسيط وسهل التطبيق.

· يضمن التحكم في FWER عند المستوى المطلوب.

عيوب بونفيروني:

Advantages of Bonferroni:

· صارم جداً، خاصة مع عدد كبير من الاختبارات.

· Simple and easy to apply.

· يزيد من خطر السلبيات الكاذبة (تفويت تأثيرات حقيقية).

· Guarantees FWER control at the desired level.

· يفترض أن الاختبارات مستقلة، وهو افتراض غير واقعي في البيانات الجينومية (الجينات مرتبطة).

Disadvantages of Bonferroni:

· Very strict, especially with a large number of tests.

· Increases the risk of false negatives (missing real effects).

· Assumes tests are independent, which is unrealistic in genomic data (genes are correlated).

```
# تصحيح بونفيروني
n_tests = 10000
alpha = 0.05
bonferroni_alpha = alpha / n_tests
print(f"مستوى الدلالة المعدل (بونفيروني): {bonferroni_alpha:.6f}")
# النتائج: 0.000005
```

3. False Discovery Rate (FDR)

The False Discovery Rate (FDR) is the proportion of false positives among all tests where the null hypothesis was rejected (i.e., genes detected as differentially expressed). In other words, FDR is the expected proportion of false positives among positive results.

3. معدل الاكتشاف الخاطئ (FDR)

معدل الاكتشاف الخاطئ (FDR) هو نسبة الإيجابيات الكاذبة بين جميع الاختبارات التي رُفضت فيها الفرضية العدمية (أي الجينات المكتشفة كذات تعبير تفاضلي). بعبارة أخرى، FDR هو النسبة المتوقعة من الإيجابيات الكاذبة بين النتائج الإيجابية.

Language / نوع الكود: **Text / FDR Example**

مثال:

إذا اكتشفنا 100 جين كذات تعبير تفاضلي، وكان FDR = 0.05، فهذا يعني أن حوالي 5 من هذه الجينات (5%) قد تكون إيجابيات كاذبة.

طريقة Benjamini-Hochberg (BH) هي الطريقة الأكثر شيوعاً للتحكم في FDR. تعمل كالتالي:

· رتب قيم p تصاعدياً.

The Benjamini-Hochberg (BH) method is the most common method for controlling FDR. It works as follows:

· ابحث عن أكبر قيمة p حيث $p(i) \le (i/m) \times \alpha$.

· Sort p-values in ascending order.

· ارفض الفرضيات العدمية لجميع الجينات ذات قيم p الأصغر أو المساوية.

· Find the largest p-value where $p(i) \le (i/m) \times \alpha$.

· Reject the null hypotheses for all genes with p-values smaller or equal.

مزايا FDR:

- أقل صرامة من FWER، مما يزيد من القدرة على اكتشاف التأثيرات الحقيقية.

- مناسب للدراسات الاستكشافية حيث يكون عدد الاختبارات كبيراً.

- يوفر توازناً بين اكتشاف الإشارات الحقيقية والتحكم في الإيجابيات الكاذبة.

عيوب FDR:

- لا يضمن عدم وجود أي إيجابيات كاذبة (على عكس FWER).

- قد يكون غير مناسب للدراسات التأكيدية حيث يكون أي خطأ من النوع الأول مكلفاً.

Advantages of FDR:

· Less strict than FWER, increasing the power to detect real effects.

· Suitable for exploratory studies where the number of tests is large.

· Provides a balance between detecting real signals and controlling false positives.

Disadvantages of FDR:

· Does not guarantee zero false positives (unlike FWER).

· May not be suitable for confirmatory studies where any Type I error is costly.

```
# محاكاة تصحيح Benjamini-Hochberg
from statsmodels.stats.multitest import multipletests
# عشوائية (محاكاة) p توليد قيم
np.random.seed(42)
p_values = np.random.uniform(0, 1, 10000)
# تطبيق تصحيح FDR (Benjamini-Hochberg)
# عدد الإيجابيات المتوقعة تحت H₀ (محاكاة)
expected_fp = sum(p_values < 0.05)  # بدون تصحيح
actual_fp = sum(rejected)  # بعد تصحيح FDR
```

print(f"الإيجابيات المتوقعة بدون تصحيح: {expected_fp}")

```
print(f"الإيجابيات بعد تصحيح FDR: {actual_fp}")
```

4. FWER vs. FDR: When to Use Which?

4. FWER مقابل FDR: متى تستخدم كل منهما؟

The choice between FWER and FDR depends on the scientific question and context:

اختيار بين FWER و FDR يعتمد على السؤال العلمي والسياق:

استخدم FWER عندما:

· أي خطأ من النوع الأول (إيجابية كاذبة) قد يكون مكلفاً أو خطيراً.

Use FWER when:

· الدراسة تأكيدية (مثل تجربة سريرية، أو قرار تنظيمي).

· Any Type I error (false positive) could be costly or dangerous.

· عدد الاختبارات صغير نسبياً.

· The study is confirmatory (e.g., clinical trial, regulatory decision).

· The number of tests is relatively small.

استخدم FDR عندما:

· الدراسة استكشافية (مثل اكتشاف جينات جديدة، أو توليد فرضيات).

Use FDR when:

· عدد الاختبارات كبير (مثل بيانات الجينوم).

· The study is exploratory (e.g., discovering new genes, generating hypotheses).

· الهدف هو تحديد مجموعة من المرشحين لمزيد من الدراسة، وليس اتخاذ قرار نهائي.

· The number of tests is large (e.g., genomic data).

· The goal is to identify a set of candidates for further study, rather than making a final decision.

مبدأ: FWER للقرارات النهائية، FDR للاستكشاف.

5. Multiple Testing Correction in RNA-seq Analysis

In RNA-seq analysis, multiple testing correction is a mandatory step. Most packages (such as DESeq2 and edgeR) automatically calculate adjusted p-values (usually using the Benjamini-Hochberg method). These adjusted values are what should be used to identify differentially expressed genes, not the raw values.

5. تصحيح الاختبارات المتعددة في تحليل RNA-seq

في تحليل RNA-seq، تصحيح الاختبارات المتعددة خطوة إلزامية. معظم الحزم (مثل DESeq2 و edgeR) تحسب تلقائياً قيم p المعدلة (عادةً باستخدام طريقة Benjamini-Hochberg). هذه القيم المعدلة هي ما يجب استخدامه لتحديد الجينات ذات التعبير التفاضلي، وليس القيم الخام.

```
# تطبيق تصحيح FDR
# مثال: تصحيح في تحليل RNA-seq محاكاة
# من تحليل DESeq2 توليد قيم p محاكاة
np.random.seed(42)
p_values = np.random.uniform(0, 1, 1000)
# عدد الجينات المكتشفة
```

```
n_sig = sum(rejected)
print(f"عدد الجينات المكتشفة (FDR < 0.05): {n_sig}")
Rule: use adjusted p-values to identify differentially expressed genes. Typically, we use an FDR threshold of 0.05 or 0.1, and report the threshold used.
```

القاعدة: استخدم القيم المعدلة (adjusted p-values) لتحديد الجينات ذات التعبير التفاضلي. عادةً، نستخدم عتبة FDR = 0.05 أو 0.1، ونبلغ عن العتبة المستخدمة.

6. Practical Application: Multiple Testing Correction

Let us apply multiple testing correction to simulated data containing real and random effects.

6. التطبيق العملي: تصحيح الاختبارات المتعددة

لنطبق تصحيح الاختبارات المتعددة على بيانات محاكاة تحتوي على تأثيرات حقيقية وتأثيرات عشوائية.

```
# محاكاة: 1000 جين، 50 منها مختلفة حقيقياً
np.random.seed(42)
n_genes = 1000
n_real = 50
# تصحيح Bonferroni
bonferroni_alpha = 0.05 / n_genes
bonferroni_sig = p_values < bonferroni_alpha
# توليد قيم p: 50 جين مع تأثير حقيقي (p صغيرة)، والباقي عشوائي
p_values = np.random.uniform(0.4, 0.9, n_genes)
p_values[:n_real] = np.random.uniform(0.001, 0.04, n_real)
# تصحيح FDR (Benjamini-Hochberg)
rejected_fdr, p_adj_fdr, _, _ = multipletests(p_values, alpha=0.05, method='fdr_bh')
# النتائج
print(f"عدد الجينات المكتشفة (بدون تصحيح، p < 0.05): {sum(p_values < 0.05)}")
print(f"عدد الجينات المكتشفة (Bonferroni): {sum(bonferroni_sig)}")
print(f"عدد الجينات المكتشفة (FDR): {sum(rejected_fdr)}")
print(f"عدد الجينات الحقيقية المختلفة: {n_real}")
# تحليل النتائج
# بدون تصحيح: العديد من الإيجابيات الكاذبة (العدد كبير)
# Bonferroni: صارم جداً، قد يفوت بعض الجينات الحقيقية
# FDR: توازن جيد بين الاكتشاف والتحكم في الإيجابيات الكاذبة
```

This example shows how different correction methods affect the number of genes detected, and why FDR is often the preferred choice in exploratory studies.

هذا المثال يوضح كيف تؤثر طرق التصحيح المختلفة على عدد الجينات المكتشفة، ولماذا FDR غالباً ما يكون الخيار المفضل في الدراسات الاستكشافية.

7. Limits of Multiple Testing Correction

Multiple testing correction is a powerful tool, but it has limits:

7. حدود تصحيح الاختبارات المتعددة

تصحيح الاختبارات المتعددة أداة قوية، لكن له حدود:

· لا يعوض عن تصميم تجربة سيء: إذا كانت العينات غير ممثلة أو المكررات غير كافية، فلن يصلح التصحيح ذلك.

· يعتمد على افتراضات: طرق مثل Bonferroni تفترض الاستقلال، وهو افتراض غير واقعي في البيانات الجينومية.

· It does not compensate for poor experimental design: if samples are unrepresentative or replicates insufficient, correction will not fix that.

· قد يزيد من السلبيات الكاذبة: الطرق الصارمة (مثل Bonferroni) قد تفوت تأثيرات حقيقية.

· Depends on assumptions: methods like Bonferroni assume independence, an unrealistic assumption in genomic data.

· التفسير يعتمد على السياق: قيمة FDR = 0.05 لا تعني أن 5% من الجينات المكتشفة خاطئة بالضرورة؛ إنها قيمة متوقعة.

· May increase false negatives: strict methods (like Bonferroni) may miss real effects.

· Interpretation depends on context: an FDR of 0.05 does not mean that 5% of the detected genes are necessarily false; it is an expected value.

مبدأ: استخدم التصحيح كأداة لدعم الاستنتاج، وليس كحكم مطلق.

Chapter Summary

خلاصة الفصل

The multiple comparisons problem arises when we test thousands of hypotheses simultaneously, increasing the probability of false positives. Family-Wise Error Rate (FWER) control using Bonferroni correction guarantees zero false positives, but is very strict and may increase false negatives. The False Discovery Rate (FDR) using the Benjamini-Hochberg method controls the proportion of false positives among positive results, and is more suitable for exploratory studies like RNA-seq analysis. The choice of method depends on the scientific question: FWER for final

decisions, FDR for exploration. Correction is a mandatory step in genomic data analysis, and the method and threshold used should be reported. The next chapter will move to Part 13 — Database Management and Scientific Resources.

مشكلة المقارنات المتعددة تنشأ عندما نختبر آلاف الفرضيات في وقت واحد، مما يزيد من احتمال الإيجابيات الكاذبة. التحكم في معدل الخطأ العائلي (FWER) باستخدام تصحيح Bonferroni يضمن عدم وجود أي إيجابيات كاذبة، لكنه صارم جداً وقد يزيد من السلبيات الكاذبة. معدل الاكتشاف الخاطئ (FDR) باستخدام طريقة Benjamini-Hochberg يتحكم في نسبة الإيجابيات الكاذبة بين النتائج الإيجابية، وهو أكثر ملاءمة للدراسات الاستكشافية مثل تحليل RNA-seq. اختيار الطريقة يعتمد على السؤال العلمي: FWER للقرارات النهائية، FDR للاستكشاف. التصحيح خطوة إلزامية في تحليل البيانات الجينومية، ويجب الإبلاغ عن الطريقة والعتبة المستخدمة. الفصل القادم سينتقل إلى الباب 13 — إدارة قواعد البيانات والمصادر العلمية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Multiple Comparisons" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "المقارنات المتعددة" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "Bonferroni correction is always the best." This is incorrect; Bonferroni is suitable for some contexts but is too strict for genomic data. The correct statement is: "the choice of multiple testing correction method depends on the scientific question and study context; Bonferroni is suitable for confirmatory studies where any Type I error is costly, while FDR is more suitable for exploratory studies." A simple test: Practical check: "Is this study confirmatory or exploratory?".

مثال على التجاوز: القول إن "تصحيح Bonferroni هو الأفضل دائماً". هذا غير صحيح؛ Bonferroni مناسب لبعض السياقات لكنه صارم جداً للبيانات الجينومية. الصحيح هو: "اختيار طريقة تصحيح الاختبارات المتعددة يعتمد على السؤال العلمي وسياق الدراسة؛ Bonferroni مناسب للدراسات التأكيدية حيث أي خطأ من النوع الأول مكلف، بينما FDR أكثر ملاءمة للدراسات الاستكشافية." اختبار بسيط: فحص تطبيقي: "هل هذه الدراسة تأكيدية أم استكشافية؟".

Example: Adjusted p-values are a computational representation of multiple testing correction. Claiming that "genes with FDR < 0.05 are the genes that cause the disease" is a biological claim not proven by the correction alone. The correction controls false positives, but proving causal roles requires additional evidence. Practical test: before inferring a biological role from a list of corrected genes,

Practical check: "Is there independent evidence supporting the role of these genes?".

مثال: قيم p المعدلة (adjusted p-values) هي تمثيل حسابي لتصحيح الاختبارات المتعددة. الادعاء بأن "الجينات ذات FDR < 0.05 هي الجينات المسببة للمرض" هو ادعاء بيولوجي لا يثبته التصحيح وحده. التصحيح يتحكم في الإيجابيات الكاذبة، لكن إثبات الدور السببي يحتاج إلى أدلة إضافية. اختبار عملي: قبل أن تستنتج دوراً بيولوجياً من قائمة جينات مصححة، فحص تطبيقي: "هل هناك أدلة مستقلة تدعم دور هذه الجينات؟"ـ

مثال تطبيقي على تصحيح FDR:

· المدخلات: قائمة بقيم p الخام، مستوى FDR المطلوب (α)ـ

· الافتراضات: أن قيم p موزعة بشكل موحد تحت الفرضية العدمية، وأن الاختبارات مستقلة أو ذات اعتماد ضعيف (في طريقة BH)ـ

· المخرجات: قيم p معدلة (q-values)، وقرارات رفض/قبول لكل فرضيةـ

· القيود: الطريقة لا تضمن عدم وجود أي إيجابيات كاذبة (على عكس FWER)، وتعتمد على افتراضات قد لا تنطبق دائماًـ

Applied example of FDR correction:

اختبار: قبل تطبيق تصحيح FDR، تحقق من توزيع قيم p، وتأكد من أن طريقة BH مناسبة لبياناتك (خاصة فيما يتعلق بالاعتماد بين الاختبارات)ـ

· Inputs: list of raw p-values, desired FDR level (α).

· Assumptions: that p-values are uniformly distributed under the null hypothesis, and that tests are independent or weakly dependent (in the BH method).

· Outputs: adjusted p-values (q-values), and reject/accept decisions for each hypothesis.

· Limitations: the method does not guarantee zero false positives (unlike FWER), and depends on assumptions that may not always hold.

Test: before applying FDR correction, check the distribution of p-values, and ensure that the BH method is suitable for the project’s data (especially regarding dependence between tests).

عند تطبيق تصحيح الاختبارات المتعددة، يجب تسجيل:

· المصدر: البيانات المستخدمة، وقيم p الخامـ

· الإصدار: إصدار الأدوات المستخدمة (مثل R، scipy، statsmodels).

· المعلمات: مستوى الدلالة (α)، طريقة التصحيح (مثل BH، Bonferroni).

· التحويلات: أي تحويلات طبقت على البيانات قبل التصحيح.

When applying multiple testing correction, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة نفس التصحيح والحصول على نفس النتائج؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the data used, and raw p-values.

· Version: the tool versions used (e.g., statsmodels, scipy, R).

· Parameters: the significance level (α), and the correction method (e.g., Bonferroni, BH).

· Transformations: any transformations applied to the data before correction.

Test: Practical check: "If I gave this record to a colleague a year later, could they reapply the same correction and obtain the same results?" If the answer is no, the documentation is insufficient.

قبل تطبيق تصحيح الاختبارات المتعددة على بيانات حقيقية، اختبر الطريقة على بيانات اصطناعية ذات قيم p معروفة. أنشئ بيانات تحت الفرضية العدمية (جميع الفرضيات صحيحة) وتحت الفرضية البديلة (بعض الفرضيات خاطئة). تحقق من أن طريقة التصحيح تتصرف كما هو متوقع (FWER أو FDR عند المستوى المطلوب). هذا يساعدك على فهم سلوك الطريقة والتحقق من صحتها.

Before applying multiple testing correction to real data, test the method on synthetic data with known p-values. Create data under the null hypothesis (all hypotheses true) and under the alternative (some hypotheses false). Verify that the correction method behaves as expected (FWER or FDR at the desired level). This helps you understand the method's behavior and verify its validity.

Part 13 — Database Management and Scientific Resources

الباب 13 — إدارة قواعد البيانات والمصادر العلمية

البحث العلمي: الفكرة المركزية، primary sources، metadata، identifiers.

Chapter 52 — Searching PubMed and Biological Databases

الفصل 52 — البحث في PubMed وقواعد البيانات

شرح أعمق للمفهوم

Deeper Conceptual Explanation

البحث العلمي نفسه عملية قابلة لإعادة الإنتاج. صياغة الكلمات المفتاحية، المرشحات، تاريخ البحث، المعرفات، والنسخ المحفوظة للمصادر كلها تؤثر في ما سيظهر. وجود PMID أو نتيجة بحث لا يعني أن الورقة تثبت الادعاء؛ يجب قراءة المصدر وتقييم تصميمه ودليله.

Scientific searching is itself a reproducible process. Search terms, filters, date, identifiers, and saved versions of sources affect what is retrieved. A PMID or search result does not mean that a paper proves a claim; the source must be read and its design and evidence evaluated.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

1. Why Systematic Scientific Search؟

1. لماذا البحث العلمي المنهجي؟

Scientific search is not just random Googling. In science, search must be systematic and reproducible. This means:

البحث العلمي ليس مجرد بحث عشوائي في Google. في العلوم، يجب أن يكون البحث منهجياً وقابلاً للتكرار. هذا يعني:

· تحديد السؤال بوضوح: ما الذي تريد معرفته بالضبط؟

· اختيار المصادر المناسبة: أي قواعد بيانات أو مجلات هي الأكثر صلة؟

· Clearly defining the question: what exactly do you want to know?

· استخدام استراتيجية بحث: كيف تبحث؟ ما هي الكلمات المفتاحية، والعوامل المنطقية (AND، OR، NOT)؟

· Choosing appropriate sources: which databases or journals are most relevant?

· توثيق البحث: ما الذي بحثت عنه، ومتى، وفي أي قاعدة بيانات؟

· Using a search strategy: how do you search? What keywords, Boolean operators (AND, OR, NOT)?

· تقييم النتائج: هل المصدر موثوق؟ هل هو دراسة أولية أم مراجعة؟ هل تمت مراجعته من قبل الأقران؟

· Documenting the search: what did you search, when, and in which database?

· Evaluating results: is the source reliable? Is it a primary study or a review? Was it peer-reviewed?

مبدأ: البحث العلمي الجيد يبدأ بسؤال جيد، ويتبع استراتيجية واضحة، ويوثق كل خطوة.

2. Formulating a Research Question

2. صياغة سؤال بحث

A good scientific question is the foundation of good research. A question like "what genes are associated with cancer?" is too broad. A better question: "which genes are differentially expressed between breast cancer cells and healthy cells in RNA-seq studies published between 2020 and 2025?" This question is specific and searchable.

السؤال العلمي الجيد هو أساس البحث الجيد. سؤال مثل "ما هي الجينات المرتبطة بالسرطان؟" واسع جداً. سؤال أفضل: "ما هي الجينات التي يختلف تعبيرها بين خلايا سرطان الثدي والخلايا السليمة في دراسات RNA-seq المنشورة بين 2020 و 2025؟" هذا السؤال محدد وقابل للبحث.

معايير السؤال الجيد (PICO):

- Population (المجتمع): سرطان الثدي.

- Intervention (التدخل): مقارنة التعبير الجيني.

- Comparison (المقارنة): خلايا سرطانية مقابل سليمة.

- Outcome (النتيجة): قائمة الجينات ذات التعبير التفاضلي.

```
# مثال: ترجمة سؤال بحث إلى استعلام في PubMed (محاكاة)
# السؤال: ما هي الجينات المرتبطة بمقاومة الإنسولين في دراسات RNA-seq؟
query = "(insulin resistance) AND (RNA-seq) AND (gene expression)"
# قد يكون الاستعلام أكثر تعقيداً مع استخدام MeSH terms، في PubMed
```

3. Information Sources: Primary Studies vs. Reviews

It is important to distinguish between two main types of scientific sources:

3. مصادر المعلومات: الدراسات الأولية مقابل المراجعات

Primary Studies: original research that presents new data. Includes laboratory experiments, clinical studies, RNA-seq analyses, etc. These are the sources that provide primary evidence.

من المهم التمييز بين نوعين رئيسيين من المصادر العلمية:

Reviews: articles that summarize and interpret existing primary studies. They do not present new data, but provide a comprehensive perspective on a specific topic. Reviews are useful for understanding the current state of knowledge, but they are not sources of primary evidence.

الدراسات الأولية (Primary Studies) : هي الأبحاث الأصلية التي تقدم بيانات جديدة. تشمل التجارب المعملية، الدراسات السريرية، تحليلات RNA-seq، إلخ. هذه هي المصادر التي تقدم الأدلة الأولية.

المراجعات (Reviews) : هي مقالات تلخص وتفسر الدراسات الأولية الموجودة. لا تقدم بيانات جديدة، بل تقدم وجهة نظر شاملة حول موضوع معين. المراجعات مفيدة لفهم الحالة الراهنة للمعرفة، لكنها ليست مصادر للأدلة الأولية.

قاعدة: استخدم المراجعات لفهم السياق، واستخدم الدراسات الأولية كأدلة أساسية.

عند الاستشهاد، استشهد بالدراسة الأولية التي قدمت البيانات، وليس بالمراجعة التي ذكرتها (ما لم تكن تناقش وجهة نظر المراجعة نفسها).

4. Persistent Identifiers: DOI and PMID

Persistent identifiers are a way to uniquely and permanently identify scientific sources. The most important are:

4. المعرفات الدائمة: DOI و PMID

المعرفات الدائمة هي طريقة لتحديد المصادر العلمية بشكل فريد ودائم. أهمها:

DOI (Digital Object Identifier): معرف رقمي فريد لكل منشور علمي. لا يتغير بمرور الوقت، ويوصل مباشرة إلى المصدر. مثال: 10.1038/s41586-020-2004-1.

PMID (PubMed Identifier): معرف فريد في قاعدة بيانات PubMed. مثال: 32641830.

> لماذا المعرفات الدائمة مهمة؟
>
> - تضمن الوصول إلى المصدر الصحيح.
>
> - تسمح بالاستشهاد الدقيق.

- تدعم إعادة الإنتاج (يمكن للآخرين العثور على نفس المصدر).

```
# مثال: استخدام DOI (محاكاة) للوصول إلى مصدر
doi = "10.1038/s41586-020-2004-1"
# يمكن استخدام APIs مثل Crossref للوصول إلى معلومات المصدر في الواقع،
```

5. Recording Versions and Access Dates for Dynamic Resources

Some sources are not static. Databases (such as GenBank) change constantly: new sequences are added, errors are corrected, and versions are updated. If you used a database in the project's analysis, you must record:

5. تسجيل الإصدار وتاريخ الوصول للموارد الديناميكية

بعض المصادر ليست ثابتة. قواعد البيانات (مثل GenBank) تتغير باستمرار: تُضاف تسلسلات جديدة، وتُصحح الأخطاء، وتُحدث الإصدارات. إذا استخدمت قاعدة بيانات في تحليلك، يجب تسجيل:

· Version: which version of the database you used (e.g., RefSeq 2025-01-15).

· الإصدار: أي إصدار من قاعدة البيانات استخدمت (مثل RefSeq 2025-01-15).

· Access date: when you accessed the database (because content may change later).

· تاريخ الوصول: متى وصلت إلى قاعدة البيانات (لأن المحتوى قد يتغير لاحقاً).

· Query: what query you used to obtain the data.

· الاستعلام: ما هو الاستعلام الذي استخدمته للحصول على البيانات.

```
# مثال: توثيق استخدام قاعدة بيانات
database_record = {
    "name": "GenBank",
    "version": "2025-01-15",
    "access_date": "2025-03-20",
    "query": "BRCA1 AND Homo sapiens",
    "url": "https://www.ncbi.nlm.nih.gov/genbank/"
}
```

This documentation is part of reproducibility. If someone else tries to repeat the project's analysis a year later, they may find different contents if you did not record the version and access date.

6. Distinguishing Primary and Secondary Sources

هذا التوثيق هو جزء من إعادة الإنتاج. إذا حاول شخص آخر تكرار تحليلك بعد عام، قد يجد محتويات مختلفة إذا لم يسجل الإصدار وتاريخ الوصول.

A primary source is the original source of information (e.g., the scientific study where the data was first published). A secondary source is a source that cites or summarizes the primary source (e.g., a review, textbook, or news article). In scientific research, primary sources should be relied upon whenever possible.

6. التمييز بين المصدر الأولي والثانوي

المصدر الأولي هو المصدر الأصلي للمعلومات (مثل الدراسة العلمية التي نُشرت فيها البيانات لأول مرة). المصدر الثانوي هو مصدر يستشهد أو يلخص المصدر الأولي (مثل مراجعة، أو كتاب مدرسي، أو مقالة إخبارية). في البحث العلمي، يجب الاعتماد على المصادر الأولية كلما أمكن.

مثال:

- المصدر الأولي: دراسة RNA-seq نشرت في Nature.

- المصدر الثانوي: مراجعة في Annual Review of Genetics تستشهد بالدراسة.

عند الاستشهاد، استشهد بالدراسة الأولية (Nature)، وليس بالمراجعة.

Exception: when the workflow is discussing the review's own perspective, or when you want a general reference to a field, the workflow can cite the review.

7. Practical Application: Searching PubMed

الاستثناء: عندما تناقش وجهة نظر المراجعة نفسها، أو عندما تريد إشارة عامة إلى مجال، يمكن الاستشهاد بالمراجعة.

In practice, searching PubMed requires a clear strategy. Here are basic steps:

7. التطبيق العملي: البحث في PubMed

في الممارسة العملية، البحث في PubMed يتطلب استراتيجية واضحة. إليك خطوات أساسية:

الخطوة الأولى: صياغة السؤال

حدد سؤالك باستخدام PICO أو إطار مشابه.

الخطوة الثانية: اختيار الكلمات المفتاحية.

اختر الكلمات المفتاحية المناسبة، واستخدم MeSH terms (المصطلحات الموحدة في PubMed) للحصول على نتائج أكثر دقة.

الخطوة الثالثة: استخدام العوامل المنطقية.

استخدم AND لتضييق النتائج، OR لتوسيعها، NOT لاستبعاد.

الخطوة الرابعة: تصفية النتائج

Step One: Formulate the question.

استخدم مرشحات مثل نوع المقالة (مراجعة، دراسة أولية)، التاريخ، اللغة.

Define the project’s question using PICO or a similar framework.

الخطوة الخامسة: توثيق البحث

Step Two: Choose keywords

سجل استعلامك، والتاريخ، وقاعدة البيانات، وعدد النتائج.

Choose appropriate keywords, and use MeSH terms (standardized terms in PubMed) for more accurate results.

Step Three: Use Boolean operators.

Use AND to narrow results, OR to broaden, NOT to exclude.

Step Four: Filter results

Use filters such as article type (review, primary study), date, language.

Step Five: Document the search

Record the project’s query, date, database, and number of results.

```
# محاكاة استعلام PubMed
search_record = {
    "database": "PubMed",
    "query": '(breast cancer) AND (RNA-seq) AND (gene expression) AND (2020:2025[pdat])',
    "filters": "primary studies, humans, English",
    "access_date": "2025-03-20",
    "results_count": 345
}
print("سجل البحث:", search_record)
```

8. Limits of Searching Scientific Sources

Searching scientific sources has limits:

8. حدود البحث في المصادر العلمية

البحث في المصادر العلمية له حدود:

· تغطية قواعد البيانات: ليست جميع المجلات مفهرسة في PubMed أو قواعد البيانات الأخرى.

· الوصول: بعض المصادر قد تكون مدفوعة أو غير متاحة.

· Database coverage: not all journals are indexed in PubMed or other databases.

· التحيز في النشر: قد تُنشر الدراسات الإيجابية أكثر من السلبية (تحيز النشر).

· Access: some sources may be paid or unavailable.

· تحديث المعرفة: المعرفة تتغير باستمرار؛ ما كان صحيحاً قبل 10 سنوات قد يكون قديماً الآن.

· Publication bias: positive studies may be published more than negative ones (publication bias).

· جودة المصادر: ليست جميع المصادر ذات جودة عالية؛ يجب تقييم كل مصدر.

· Knowledge updates: knowledge changes constantly; what was true 10 years ago may be outdated now.

· Source quality: not all sources are of high quality; each source must be evaluated.

قاعدة: تحقق دائماً من مصداقية المصدر. هل هو من مجلة محكمة؟ هل المؤلفون معروفون في المجال؟ هل

تمت مراجعته من قبل الأقران؟

9. Practical Application: Evaluating a Source

Before using any source, evaluate it using criteria such as:

9. التطبيق العملي: تقييم مصدر

قبل استخدام أي مصدر، قم بتقييمه باستخدام معايير مثل:

· المصداقية: هل المصدر من مجلة علمية محكمة؟ هل المؤلفون لديهم خبرة في المجال؟

· الدقة: هل البيانات والاستنتاجات مدعومة بأدلة؟ هل توجد أخطاء واضحة؟

· Credibility: is the source from a peer-reviewed scientific journal? Do the authors have expertise in the field?

· الموضوعية: هل هناك تضارب في المصالح؟ هل الدراسة ممولة من جهة قد تؤثر على النتائج؟

· Accuracy: are the data and conclusions supported by evidence? Are there obvious errors?

· التوقيت: هل المصدر حديث بما يكفي لسؤالك؟ هل هناك دراسات أحدث؟

· Objectivity: is there a conflict of interest? Is the study funded by a party that might affect the results?

· الصلة: هل المصدر مرتبط مباشرة بسؤالك؟

· Timeliness: is the source recent enough for the project’s question? Are there more recent studies?

· Relevance: is the source directly related to the project’s question?

```
# محاكاة تقييم مصدر
source_evaluation = {
    "title": "RNA-seq analysis of breast cancer gene expression",
    "journal": "Nature",
    "peer_reviewed": True,
    "authors": ["Smith J", "Johnson K"],
    "year": 2023,
    "conflict_of_interest": "None declared",
    "relevance": "High",
    "decision": "Use as primary source"
}
print("تقييم المصدر:", source_evaluation)
```

Chapter Summary

خلاصة الفصل

Systematic scientific search is the foundation of every reliable analysis. It requires formulating a clear research question, choosing appropriate sources, using a systematic search strategy, and documenting every step. Distinguishing primary studies from reviews is essential: primary studies are sources of evidence, and reviews provide context. Persistent identifiers (DOI, PMID) ensure access to the correct sources. Recording versions and access dates for dynamic resources is part of reproducibility. Evaluating sources for credibility, accuracy, objectivity, timeliness, and relevance is the researcher's responsibility. The next chapter will move to local datasets versus live databases.

البحث العلمي المنهجي هو أساس كل تحليل موثوق. يتطلب صياغة سؤال بحث واضح، واختيار المصادر المناسبة، واستخدام استراتيجية بحث منهجية، وتوثيق كل خطوة. التمييز بين الدراسات الأولية والمراجعات هو أساسي: الدراسات الأولية هي مصادر الأدلة، والمراجعات تقدم سياقاً. المعرفات الدائمة (DOI، PMID) تضمن الوصول إلى المصادر الصحيحة. تسجيل الإصدار وتاريخ الوصول للموارد الديناميكية هو جزء من إعادة الإنتاج. تقييم المصادر من حيث المصداقية والدقة والموضوعية والتوقيت والصلة هو مسؤولية الباحث. الفصل القادم سينتقل إلى البيانات المحلية مقابل البيانات الحية.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Searching PubMed and Biological Databases" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "البحث في PubMed وقواعد البيانات" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "searching PubMed answers all scientific questions." This is incorrect; PubMed is a tool, but it does not contain all knowledge, and its coverage may not include some fields. The correct statement is: "searching PubMed is a powerful tool for accessing scientific literature in medicine and biology, but it is not comprehensive, and researchers may need other databases depending on their field." A simple test: Practical check: "Is everything I need available in PubMed?".

مثال على التجاوز: القول إن "البحث في PubMed يجيب على جميع الأسئلة العلمية". هذا غير صحيح؛ PubMed هو أداة، لكنه لا يحتوي على جميع المعرفة، وقد لا تغطي قاعدته بعض المجالات. الصحيح هو: "البحث في PubMed هو أداة قوية للوصول إلى الأدبيات العلمية في مجالات الطب والبيولوجيا، لكنه ليس شاملاً، وقد يحتاج الباحث إلى قواعد بيانات أخرى حسب مجاله." اختبار بسيط: فحص تطبيقي: "هل كل ما أحتاجه موجود في PubMed؟"ـ

Example: The presence of an article in PubMed with a PMID is a computational representation of the source's existence. Claiming that "the article proves a causal relationship" is a biological claim not proven by the mere existence of the article. The article must be read and critically evaluated. Practical test: before inferring something from an article, Practical check: "Have I read the full article? Have I evaluated its quality?".

مثال: وجود مقال في PubMed مع PMID هو تمثيل حسابي لوجود المصدر. الادعاء بأن "المقال يثبت وجود علاقة سببية" هو ادعاء بيولوجي لا يثبته مجرد وجود المقال. المقال يجب قراءته وتقييمه نقدياً. اختبار عملي: قبل أن تستنتج شيئاً من مقال، فحص تطبيقي: "هل قرأت المقال كاملاً؟ هل قيمت جودته؟"ـ

مثال تطبيقي على البحث في PubMed:

· المدخلات: سؤال بحث، كلمات مفتاحية، مرشحات (نوع المقال، التاريخ، اللغة)ـ

· الافتراضات: أن قواعد البيانات تحتوي على المصادر ذات الصلة، وأن المرشحات المستخدمة مناسبةـ

· المخرجات: قائمة بالمقالات ذات الصلة، مع المعرفات والملخصاتـ

· القيود: قد لا تحتوي قواعد البيانات على جميع المصادر، وقد تكون بعض المصادر مدفوعة أو غير متاحةـ

Applied example of searching PubMed:

اختبار: قبل الاعتماد على نتائج البحث، تحقق من تغطية قاعدة البيانات، وتأكد من أن استراتيجية البحث شاملةـ

· Inputs: research question, keywords, filters (article type, date, language).

· Assumptions: that the databases contain relevant sources, and that the filters used are appropriate.

· Outputs: a list of relevant articles, with identifiers and abstracts.

· Limitations: databases may not contain all sources, and some sources may be paid or unavailable.

Test: before relying on search results, check database coverage, and ensure the search strategy is comprehensive.

عند البحث في المصادر العلمية، يجب تسجيل:

· المصدر: قاعدة البيانات المستخدمة، والمجلة، والمقال (مع DOI أو PMID).

· الإصدار: إصدار قاعدة البيانات (إذا كان متاحاً)، وتاريخ الوصول.

· المعلمات: استعلام البحث، والمرشحات المستخدمة.

· التحويلات: أي خطوات لتصفية أو تقييم المصادر.

When searching scientific sources, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع إعادة نفس البحث والعثور على نفس المصادر؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: the database used, the journal, and the article (with DOI or PMID).

· Version: the database version (if available), and access date.

· Parameters: the search query, and filters used.

· Transformations: any steps to filter or evaluate sources.

Test: Practical check: "If I gave this record to a colleague a year later, could they repeat the same search and find the same sources?" If the answer is no, the documentation is insufficient.

قبل الاعتماد على استراتيجية بحث في مشروع حقيقي، اختبرها على أسئلة تعرف إجاباتها مسبقاً. ابحث عن موضوع تعرف أنه يحتوي على مراجع معروفة، وتحقق من أن استراتيجيتك تجد تلك المراجع. إذا نجح الاختبار، فأنت واثق من أن استراتيجيتك تعمل. عند الانتقال إلى أسئلة جديدة، تذكر أن كل موضوع قد يتطلب تعديلات في الاستراتيجية.

Before relying on a search strategy in a real project, test it on questions with known answers. Search for a topic you know has known references, and verify that the project's strategy finds those references. If the test succeeds, the workflow is confident that the project's strategy works. When moving to new questions, remember that each topic may require adjustments to the strategy.

Chapter 53 — Local Datasets versus Live Databases

الفصل 53 — البيانات المحلية مقابل البيانات الحية

Deeper Conceptual Explanation

الفكرة المركزية: synthetic fixtures، stability، versioned snapshots، حدود البيانات الحية.

شرح أعمق للمفهوم

Local data are relatively stable and can be preserved with a project, while live data can change, disappear, or be reindexed. Recording only a database URL is therefore insufficient; identifiers, versions, dates, or an appropriate preserved snapshot should be recorded when permitted.

البيانات المحلية ثابتة نسبياً ويمكن حفظها مع المشروع، بينما البيانات الحية قد تتغير أو تُحذف أو تعاد فهرستها.

لذلك لا يكفي تسجيل عنوان قاعدة البيانات؛ يجب حفظ المعرفات والإصدارات والتاريخ، أو نسخة مناسبة من البيانات

عندما تسمح الشروط.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Why Are Live Data a Problem for Education and Reproducibility؟

1. لماذا البيانات الحية مشكلة للتعليم وإعادة الإنتاج؟

Live databases, such as GenBank or RefSeq, are constantly changing for legitimate reasons: new sequences are added, errors are corrected, and classifications are updated. These changes are useful for scientific research, but they pose a challenge for education and reproducibility:

قواعد البيانات الحية، مثل GenBank أو RefSeq، تتغير باستمرار لأسباب مشروعة: تضاف تسلسلات جديدة،

وتُصحح الأخطاء، وتُحدث التصنيفات. هذه التغييرات مفيدة للبحث العلمي، لكنها تشكل تحدياً للتعليم وإعادة الإنتاج:

· Education: if an educational lesson depends on an example from a live database, the example may not work a month later due to data changes or sequence removal.

· التعليم: إذا كان الدرس التعليمي يعتمد على مثال من قاعدة بيانات حية، فقد لا يعمل المثال بعد شهر بسبب تغير

البيانات أو إزالة التسلسل.

· Reproducibility: if you publish an analysis that depends on a live database without recording the version, others may not be able to reproduce the project's results a year later.

· إعادة الإنتاج: إذا نشرت تحليلاً يعتمد على قاعدة بيانات حية دون تسجيل الإصدار، فقد لا يستطيع الآخرون إعادة إنتاج نتائجك بعد عام.

· Dependency: if the network is down or the database is unavailable, the analysis cannot be run.

· الاعتمادية: إذا كانت الشبكة معطلة أو كانت قاعدة البيانات غير متاحة، فلا يمكن تشغيل التحليل.

مثال:

درس يعتمد على تسلسل معين من GenBank. بعد تحديث قاعدة البيانات، قد يتغير رقم الانضمام أو يُزال التسلسل.

الدرس يصبح غير قابل للتشغيل، ويحتاج المدرس إلى تحديثه باستمرار.

2. The Solution: Stable Synthetic Data (Synthetic Fixtures)

Stable synthetic data are small data files, created manually or by software, that represent real data but are stable and do not change. They are used for educational purposes, code testing, and concept illustration. Their advantages:

2. الحل: البيانات الاصطناعية الثابتة (Synthetic Fixtures)

البيانات الاصطناعية الثابتة هي ملفات بيانات صغيرة، مُنشأة يدوياً أو بواسطة برنامج، تمثل بيانات حقيقية لكنها ثابتة ولا تتغير. تستخدم للأغراض التعليمية، واختبار الكود، وتوضيح المفاهيم. مزاياها:

· ثبات: لا تتغير بمرور الوقت، مما يضمن تكرار المثال.

· Stability: they do not change over time, ensuring example reproducibility.

· استقلالية: لا تعتمد على اتصال بالإنترنت.

· Independence: they do not depend on internet connectivity.

· بساطة: يمكن تصميمها لتوضح حالة معينة (مثل تسلسل يحتوي على طفرة، أو عينة ذات تعبير مرتفع).

· Simplicity: they can be designed to illustrate a specific case (e.g., a sequence with a mutation, or a sample with high expression).

· أمان: لا تحتوي على بيانات حساسة أو حقيقية، مما يسمح بمشاركتها بحرية.

· Safety: they do not contain sensitive or real data, allowing free sharing.

```
# مثال: بيانات اصطناعية في ملف FASTA (محاكاة)
synthetic_fasta = """
>SYN_001 synthetic sequence for testing
ATGCGTACGTAGCTAGCTAGCTAGCTAGC
>SYN_002 another synthetic sequence
GCTAGCTAGCTAGCTAGCTAGCTAGCTA
"""
# حفظ الملف واستخدامه في الاختبارات التعليمية
```

مبدأ: البيانات الاصطناعية هي أداة تعليمية واختبارية، وليست بديلاً عن البيانات الحقيقية في البحث العلمي.

3. Using Synthetic Data in Educational Tests

In the educational context, synthetic data can be used to teach:

3. استخدام البيانات الاصطناعية في الاختبارات التعليمية

في السياق التعليمي، يمكن استخدام البيانات الاصطناعية لتعليم:

· قراءة الملفات: كيفية قراءة ملفات FASTA أو FASTQ باستخدام Biopython.

· File reading: how to read FASTA or FASTQ files using Biopython.

· تحليل البيانات: كيفية حساب نسبة GC، أو ترجمة تسلسل، أو محاذاة تسلسلات.

· Data analysis: how to calculate GC content, translate a sequence, or align sequences.

· التحقق من الكود: التأكد من أن الدوال تعمل كما هو متوقع قبل تطبيقها على بيانات حقيقية.

· Code verification: ensuring that functions work as expected before applying them to real data.

· فهم المفاهيم: توضيح مفاهيم مثل التطبيع، أو التعبير التفاضلي، باستخدام بيانات ذات خصائص معروفة.

· Concept understanding: illustrating concepts such as normalization or differential expression using data with known properties.

```
# مثال: اختبار دالة حساب نسبة GC على بيانات اصطناعية
def test_gc_content():
    seq = "ATGCGTAC"
    expected_gc = 50.0  # 4 قواعد هي 8 من G أو C
    calculated_gc = gc_content(seq)
    assert calculated_gc == expected_gc, f"Expected {expected_gc}, got {calculated_gc}"
    print("اختبار نسبة GC اجتاز بنجاح!")
test_gc_content()
```

الدوال المعرفة هنا (test_gc_content) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق

الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

4. Distinguishing Educational Data from Biological Evidence

It is important to understand that synthetic data is not biological evidence. It is an educational and testing tool. It cannot be used to draw real biological conclusions. When moving to scientific research, real data must be used, with source and version documentation. Synthetic data is for learning and verification, not for biological proof.

4. التمييز بين البيانات التعليمية والأدلة البيولوجية

من المهم أن نفهم أن البيانات الاصطناعية ليست أدلة بيولوجية. هي أداة تعليمية واختبارية. لا يمكن استخدامها لاستخلاص استنتاجات بيولوجية حقيقية. عند الانتقال إلى البحث العلمي، يجب استخدام البيانات الحقيقية، مع توثيق المصدر والإصدار. البيانات الاصطناعية هي للتعلم والتحقق، وليس للإثبات البيولوجي.

قاعدة:

- استخدم البيانات الاصطناعية للتعلم والاختبار.

- استخدم البيانات الحقيقية الموثقة للبحث العلمي.

- لا تخلط بينهما: نتيجة على بيانات اصطناعية لا تعني شيئاً عن الواقع البيولوجي.

5. Limits of Live Data

Relying on live data has limits that must be clear:

5. حدود البيانات الحية

الاعتماد على البيانات الحية له حدود يجب أن تكون واضحة:

· التغير المستمر: تتغير قواعد البيانات باستمرار، مما قد يؤثر على النتائج.

· التوفر: قد تكون قاعدة البيانات غير متاحة مؤقتاً (صيانة، أعطال شبكة).

· Continuous change: databases change constantly, which may affect results.

· الاعتماد على الشبكة: يتطلب اتصالاً بالإنترنت، مما قد لا يكون متاحاً دائماً.

· Availability: the database may be temporarily unavailable (maintenance, network failures).

· الإصدارات: قد تختلف النتائج بين إصدارات مختلفة من قاعدة البيانات.

· Network dependency: requires internet connectivity, which may not always be available.

· التكلفة: بعض قواعد البيانات قد تكون مدفوعة أو تتطلب اشتراكاً.

· Versions: results may differ between different versions of the database.

· Cost: some databases may be paid or require a subscription.

مبدأ: عند استخدام بيانات حية، سجل الإصدار وتاريخ الوصول. هذا هو شرط إعادة الإنتاج.

6. Practical Application: Building a Synthetic Dataset for Education

Let us apply these principles by building a small synthetic dataset for teaching RNA-seq analysis:

6. التطبيق العملي: بناء مجموعة بيانات اصطناعية للتعليم

لنطبق هذه المبادئ ببناء مجموعة بيانات اصطناعية صغيرة لتعليم تحليل RNA-seq:

Purpose: organize results in a tabular structure for inspection, saving, and downstream analysis.

الغرض: تنظيم النتائج في بنية جدوليّة تسهّل الفحص والحفظ والتحليل اللاحق.

```
# بناء بيانات اصطناعية لتعليم RNA-seq
import numpy as np
import pandas as pd
# توليد بيانات تعبير اصطناعية: 10 جينات، 6 عينات (3 سليمة، 3 مريضة)
np.random.seed(42)
n_genes = 10
n_samples = 6
# توليد أعداد قراءات عشوائية
counts = np.random.poisson(lam=50, size=(n_genes, n_samples))
# إنشاء DataFrame
gene_names = [f"GENE_{i+1}" for i in range(n_genes)]
sample_names = [f"S{i+1}" for i in range(n_samples)]
df = pd.DataFrame(counts, index=gene_names, columns=sample_names)
# جعل بعض الجينات مختلفة بين المجموعتين
for i in range(3):
# إضافة بيانات وصفية (condition)
metadata = pd.DataFrame({
"sample": sample_names,
"condition": ["healthy", "healthy", "healthy", "diseased", "diseased", "diseased"]
```

```
})
```

```
# حفظ البيانات
df.to_csv("synthetic_counts.csv")
metadata.to_csv("synthetic_metadata.csv")
```

print("تم إنشاء بيانات اصطناعية للتعليم:")

```
print(df)
print("\nالبيانات الوصفية:")
print(metadata)
```

This synthetic data can be used to teach differential expression analysis, normalization, and visualization, without relying on real data or internet connectivity.

7. Documentation and Reproducibility with Synthetic Data

هذه البيانات الاصطناعية يمكن استخدامها لتعليم تحليل التعبير التفاضلي، والتطبيع، والتصور، دون الاعتماد على

بيانات حقيقية أو اتصال بالإنترنت.

Even with synthetic data, it must be documented. Record:

7. التوثيق وإعادة الإنتاج مع البيانات الاصطناعية

حتى مع البيانات الاصطناعية، يجب توثيقها. سجل:

· كيف أُنشئت: ما هي المعلمات المستخدمة (مثل التوزيع، عدد الجينات، عدد العينات)؟

· How it was created: what parameters were used (e.g., distribution, number of genes, number of samples)?

· خصائصها: ما هي الخصائص المعروفة للبيانات (مثل عدد الجينات المختلفة)؟

· Its properties: what are the known properties of the data (e.g., number of different genes)?

· الغرض منها: لأي غرض تعليمي أو اختباري صُممت؟

· Its purpose: what educational or testing purpose was it designed for?

· الإصدار: تاريخ إنشائها، وأي تحديثات.

· Version: creation date, and any updates.

```
# مثال: توثيق البيانات الاصطناعية
synthetic_data_documentation = {
```

"name": "Synthetic RNA-seq dataset for teaching",

"created": "2025-03-20",

"parameters": {

```
"n_genes": 10,
"n_samples": 6,
"distribution": "Poisson(50)",
"differential_genes": 3,
"fold_change": 2
},
"purpose": "Teaching differential expression analysis",
"version": "1.0"
}
```


```
print("توثيق البيانات الاصطناعية:", synthetic_data_documentation)
```

8. From Synthetic Data to Real Data

Synthetic data is the starting point, not the end. After understanding concepts and verifying code using synthetic data, the workflow can move to real data. But the transition requires:

8. من البيانات الاصطناعية إلى البيانات الحقيقية

البيانات الاصطناعية هي نقطة البداية، وليست النهاية. بعد فهم المفاهيم والتحقق من الكود باستخدام البيانات الاصطناعية، يمكن الانتقال إلى البيانات الحقيقية. لكن الانتقال يتطلب:

· فهم حدود البيانات الاصطناعية: ما الذي تمثله وما لا تمثله.

· Understanding the limits of synthetic data: what it represents and what it does not.

· توثيق البيانات الحقيقية: تسجيل المصدر والإصدار وتاريخ الوصول.

· Documenting real data: recording source, version, and access date.

· التحقق من الاختلافات: قد تختلف البيانات الحقيقية عن الاصطناعية في التوزيع، والتباين، والجودة.

· Checking differences: real data may differ from synthetic in distribution, variance, and quality.

· تعديل الكود إذا لزم الأمر: قد تحتاج بعض الدوال إلى تعديل للتعامل مع تعقيدات البيانات الحقيقية.

· Modifying code if necessary: some functions may need adjustment to handle the complexities of real data.

مبدأ: البيانات الاصطناعية للتعلم، والبيانات الحقيقية للبحث.

استخدم الأولى لفهم الأدوات، والثانية للإجابة عن الأسئلة العلمية.

Chapter Summary

خلاصة الفصل

Live databases are constantly changing, posing a challenge for education and reproducibility. The solution is to use stable synthetic data (synthetic fixtures) for educational purposes and code testing. This data does not change, does not depend on internet connectivity, and can be designed to illustrate specific cases. But it is important to distinguish educational data from biological evidence: synthetic data is not a substitute for real data in scientific research. When using live data, source, version, and access date must be documented. Synthetic data is the starting point, and real data is the goal. The next chapter will move to documentation and reproducibility.

البيانات الحية تتغير باستمرار، مما يشكل تحدياً للتعليم وإعادة الإنتاج. الحل هو استخدام بيانات اصطناعية ثابتة (synthetic fixtures) للأغراض التعليمية واختبار الكود. هذه البيانات لا تتغير، ولا تعتمد على اتصال بالإنترنت، ويمكن تصميمها لتوضح حالات محددة. لكن من المهم التمييز بين البيانات التعليمية والأدلة البيولوجية: البيانات الاصطناعية ليست بديلاً عن البيانات الحقيقية في البحث العلمي. عند استخدام بيانات حية، يجب توثيق المصدر والإصدار وتاريخ الوصول. البيانات الاصطناعية هي نقطة البداية، والبيانات الحقيقية هي الهدف. الفصل القادم سينتقل إلى التوثيق وقابلية إعادة الإنتاج.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Local Datasets versus Live Databases" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "البيانات المحلية مقابل البيانات الحية" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "synthetic data can be used instead of real data in scientific research." This is incorrect; synthetic data is for education and testing, not a substitute for real data. The correct statement is: "synthetic data is an educational and testing tool that allows understanding concepts and verifying code, but it does not replace real data in scientific research, which requires precise documentation of source and version." A simple test: Practical check: "Does this data represent biological reality, or is it just a learning tool?".

مثال على التجاوز: القول إن "البيانات الاصطناعية يمكن استخدامها بدلاً من البيانات الحقيقية في البحث العلمي". هذا غير صحيح؛ البيانات الاصطناعية للتعليم والاختبار، وليست بديلاً عن البيانات الحقيقية. الصحيح هو: "البيانات الاصطناعية هي أداة تعليمية واختبارية تسمح بفهم المفاهيم والتحقق من الكود، لكنها لا تغني عن البيانات الحقيقية في البحث العلمي، التي تتطلب توثيقاً دقيقاً للمصدر والإصدار." اختبار بسيط: فحص تطبيقي: "هل هذه البيانات تمثل الواقع البيولوجي، أم أنها مجرد أداة للتعلم؟"ـ

Example: An analysis result on synthetic data showing an expression difference is a computational representation. Claiming that "this difference exists in biological reality" is a biological claim not proven by analysis on synthetic data. Analysis on synthetic data shows that the code works, but it says nothing about reality. Practical test: before inferring something biological from an analysis, Practical check: "Is this data real or synthetic?".

مثال: نتيجة تحليل على بيانات اصطناعية تظهر اختلافاً في التعبير هي تمثيل حسابي. الادعاء بأن "هذا الاختلاف موجود في الواقع البيولوجي" هو ادعاء بيولوجي لا يثبته التحليل على بيانات اصطناعية. التحليل على بيانات اصطناعية يوضح أن الكود يعمل، لكنه لا يقول شيئاً عن الواقع. اختبار عملي: قبل أن تستنتج شيئاً بيولوجياً من تحليل، فحص تطبيقي: "هل هذه البيانات حقيقية أم اصطناعية؟"ـ

مثال تطبيقي على استخدام بيانات اصطناعية:

· المدخلات: ملفات اصطناعية (FASTA، counts، metadata)، دوال تحليلـ

· الافتراضات: أن البيانات الاصطناعية تمثل سيناريو تعليمي معقول، وأن الدوال تعمل بشكل صحيح على هذا النوع من البياناتـ

· المخرجات: نتائج تحليل (مثل قيم تطبيع، رسوم بيانية)، وتقارير اختبارـ

· القيود: النتائج لا تعكس الواقع البيولوجي، ولا يمكن استخدامها للاستدلال البيولوجيـ

Applied example of using synthetic data:

اختبار: قبل استخدام بيانات اصطناعية، تحقق من أنها تغطي الحالات التي تريد اختبارها، وافهم حدودهاـ

· Inputs: synthetic files (FASTA, counts, metadata), analysis functions.

· Assumptions: that the synthetic data represents a reasonable educational scenario, and that the functions work correctly on this type of data.

· Outputs: analysis results (e.g., normalized values, plots), and test reports.

· Limitations: the results do not reflect biological reality, and cannot be used for biological inference.

Test: before using synthetic data, verify that it covers the cases you want to test, and understand its limits.

عند إنشاء أو استخدام بيانات اصطناعية، يجب تسجيل:

· المصدر: كيف أُنشئت البيانات (المعلمات، الكود المستخدم).

· الإصدار: تاريخ الإنشاء، وأي تحديثات.

· المعلمات: المعلمات المستخدمة في التوليد (مثل عدد الجينات، التوزيع).

· التحويلات: أي تحويلات طبقت على البيانات بعد التوليد.

When creating or using synthetic data, record:

اختبار: فحص تطبيقي: "لو أعطيت هذه البيانات الاصطناعية لزميل بعد عام، هل يستطيع فهم كيف أُنشئت وإعادة توليدها؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: how the data was created (parameters, code used).

· Version: creation date, and any updates.

· Parameters: parameters used in generation (e.g., number of genes, distribution).

· Transformations: any transformations applied to the data after generation.

Test: Practical check: "If I gave this synthetic data to a colleague a year later, could they understand how it was created and regenerate it?" If the answer is no, the documentation is insufficient.

قبل الاعتماد على تحليل على بيانات حقيقية، اختبر الكود والمسار على بيانات اصطناعية. استخدم بيانات اصطناعية ذات خصائص معروفة، وتأكد من أن الكود يعطي النتائج المتوقعة. هذا يضمن أن أي مشكلة مع بيانات حقيقية تعكس خصائص البيانات وليس خطأ في الكود. عند الانتقال إلى بيانات حقيقية، تذكر أن البيانات الاصطناعية لا تغطي كل التعقيدات الواقعية.

Before relying on analysis on real data, test the code and workflow on synthetic data. Use synthetic data with known properties, and verify that the code gives the expected results. This ensures that any problem with real data reflects the data's properties, not a code error. When moving to real data, remember that synthetic data does not cover all real-world complexities.

Chapter 54 — Documentation and Reproducibility

الفصل 54 — التوثيق وقابلية إعادة الإنتاج

الفكرة المركزية: provenance, environment, parameters, logs, reproducible reports.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

Documentation is not an appendix added after analysis; it is part of how the result is produced. The record should state provenance, versions, parameters, transformations, outputs, and whether data were static or live. When results differ a year later, the record becomes an explanation tool rather than a mere archive.

التوثيق ليس ملحقاً بعد انتهاء التحليل؛ هو جزء من طريقة إنتاج النتيجة. يجب أن يوضح السجل المصدر، والإصدارات، والمعلمات، والتحويلات، والملفات الناتجة، وما إذا كانت البيانات ثابتة أم حية. عندما تختلف النتيجة بعد عام، يصبح السجل وسيلة تفسير لا مجرد أرشيف.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. Why Documentation?

Documentation is not just a luxury or an administrative requirement; it is an essential part of the scientific method. Without documentation, results cannot be verified, built upon, or traced for errors. Good documentation protects the researcher from their own forgetfulness, and protects the scientific community from relying on unverifiable results.

1. لماذا التوثيق؟

التوثيق ليس مجرد رفاهية أو متطلباً إدارياً؛ إنه جزء أساسي من المنهج العلمي. بدون توثيق، لا يمكن التحقق من النتائج، ولا يمكن بناؤها عليها، ولا يمكن تتبع الأخطاء. التوثيق الجيد يحمي الباحث من نسيانه هو نفسه، ويحمي المجتمع العلمي من الاعتماد على نتائج لا يمكن التحقق منها.

مبادئ التوثيق الجيد:

1. الشفافية: كل خطوة موثقة.

2. القابلية للتدقيق: يمكن تتبع كل قرار.

3. القابلية للتكرار: يمكن لشخص آخر إعادة التشغيل (أو على الأقل فهم ما حدث).

4. الصدق: توثيق القيود والافتراضات.

2. Components of a Reproducibility Record

A good reproducibility record should contain the following elements:

2. مكونات سجل إعادة الإنتاج

السؤال العلمي (Research Question)ـ

سجل إعادة الإنتاج الجيد يجب أن يحتوي على العناصر التالية:

ما هو السؤال الذي تحاول الإجابة عنه؟ هذا يحدد سياق التحليل ويساعد في تفسير النتائجـ

البيانات ومصدرها (Data and Provenance)ـ

من أين جاءت البيانات؟ ما هو رقم الإصدار؟ متى تم الحصول عليها؟ ما هي التحويلات التي طُبقت عليها؟

البيئة البرمجية والإصدارات (Software Environment and Versions)ـ

ما هو نظام التشغيل؟ ما هو إصدار Python (أو R)؟ ما هي إصدارات المكتبات المستخدمة (Biopython، NumPy، DESeq2)؟

خطوات المعالجة (Processing Steps)ـ

ما هي الخطوات التي طُبقت على البيانات الخام؟ بأي ترتيب؟ ما هي المعاملات المستخدمة في كل خطوة؟

المعاملات (Parameters)

ما هي العتبات المستخدمة؟ (مثل عتبة الجودة، عتبة التشابه، مستوى الدلالة)ـ

النتائج (Outputs)

القيود (Limitations)

ما هي النتائج النهائية؟ (قوائم الجينات، الرسوم البيانية، الإحصائيات)ـ

Research Question

What question are you trying to answer? This defines the context of the analysis and helps interpret the results.

ما هي حدود هذا التحليل؟ ما هي الافتراضات التي قد لا تنطبق؟ ما هو نطاق صحة النتائج؟

Data and Provenance

Where did the data come from? What is the version number? When was it obtained? What transformations were applied?

Software Environment and Versions.

What is the operating system? What is the Python (or R) version? What are the versions of the libraries used (Biopython, NumPy, DESeq2)?

Processing Steps

What steps were applied to the raw data? In what order? What parameters were used in each step?

Parameters

What thresholds were used? (e.g., quality threshold, similarity threshold, significance level).

Outputs

What are the final results? (gene lists, plots, statistics).

Limitations

What are the limits of this analysis? What assumptions may not hold? What is the scope of validity of the results?

```
# مثال: هيكل سجل إعادة الإنتاج
reproducibility_record = {
```

"research_question": "ما هي الجينات ذات التعبير التفاضلي بين خلايا سرطان الثدي والخلايا السليمة؟",

"data": {

"source": "GEO (GSE12345)",

"version": "GSE12345_2025-01-15",

"access_date": "2025-03-20",

"transformations": ["log2 transformation", "quantile normalization"]

},

"software": {

"os": "Ubuntu 22.04",

"python_version": "3.11.4",

"packages": {

"biopython": "1.81",

"numpy": "1.24.3",

"pandas": "2.0.3"

}

},

"processing": [

```
{"step": "quality_control", "tool": "FastQC", "params": {"min_quality": 20}},
{"step": "trimming", "tool": "Trimmomatic", "params": {"leading": 3, "trailing": 3}},
{"step": "alignment", "tool": "STAR", "params": {"genome": "hg38"}},
{"step": "quantification", "tool": "featureCounts", "params": {"counts": "gene"}},
{"step": "normalization", "tool": "DESeq2", "params": {"method": "median-of-ratios"}},
{"step": "differential_expression", "tool": "DESeq2", "params": {"alpha": 0.05, "lfcThreshold": 1}}
],
"outputs": {
"significant_genes": 45,
"top_genes": ["BRCA1", "TP53", "ESR1", ...],
"plots": ["PCA_plot.png", "MA_plot.png", "volcano_plot.png"]
},
"limitations": [
"حجم العينة صغير (n=6)",
"قد تكون هناك تأثيرات دفعات غير مقيّدة",
"النتائج تحتاج إلى تحقق تجريبي مستقل"
]
}
```

3. Documenting Limitations as Part of the Scientific Result

A common mistake is hiding or downplaying limitations for fear of negative evaluation. But limitations are part of the scientific result. Every analysis has limits; knowing these limits allows others to correctly interpret the results and determine whether they apply to their context. Documenting limitations shows scientific maturity and transparency.

3. توثيق القيود كجزء من النتيجة العلمية

من الأخطاء الشائعة إخفاء القيود أو التقليل من شأنها خوفاً من تقييم سلبي. لكن القيود هي جزء من النتيجة العلمية. كل تحليل له حدود؛ معرفة هذه الحدود تسمح للآخرين بتفسير النتائج بشكل صحيح وتحديد ما إذا كانت تنطبق على سياقهم. توثيق القيود يظهر نضجاً علمياً وشفافية.

أسئلة لتحديد القيود:

- ما هي الافتراضات التي انتهكتها البيانات؟

- ما هي القيم المتطرفة التي أُزيلت ولماذا؟

- هل العينة كافية للاستدلال المطلوب؟

- هل هناك عوامل مربكة لم تُقيّد؟

- ما هو نطاق صحة الاستنتاجات؟

4. Handling Changes in Sources and Versions

In the real world, sources and versions change. What is available today may not be available tomorrow. Instead of hiding these changes, they should be documented and interpreted. If a biological source, library version, or normalization method changes, the difference in output is not an error to be hidden; it is information to be interpreted. This is the essence of the scientific method: not pretending everything is fixed, but understanding how changes affect results.

4. التعامل مع التغيرات في المصادر والإصدارات

في العالم الحقيقي، تتغير المصادر والإصدارات. ما كان متاحاً اليوم قد لا يكون متاحاً غداً. بدلاً من إخفاء هذه التغيرات، يجب توثيقها وتفسيرها. إذا تغير مصدر حي أو إصدار مكتبة أو طريقة تطبيع، فاختلاف الناتج ليس خطأ يجب إخفاؤه؛ بل معلومة يجب تفسيرها. هذا هو جوهر المنهج العلمي: ليس التظاهر بأن كل شيء ثابت، بل فهم كيفية تأثير التغيرات على النتائج.

مبدأ: إذا تغيرت النتيجة بسبب تغير الإصدار، فهذه معلومة، وليس خطأ.

وثق الإصدارات وتاريخ الوصول حتى يمكن تفسير التغيرات.

5. Practical Application: Building a Reproducibility Record

Let us apply these principles to a hypothetical project:

5. التطبيق العملي: بناء سجل إعادة الإنتاج

لنطبق هذه المبادئ على مشروع وهمي:

```
# مشروع وهمي: تحليل RNA-seq
def create_reproducibility_record():
    record = {
        "project": {
            "name": "RNA-seq analysis of breast cancer",
            "date": "2025-03-20",
            "author": "researcher_01"
        },
        "question": {
            "description": "تحديد الجينات ذات التعبير التفاضلي بين خلايا سرطان الثدي والخلايا السليمة",
        },
        "data": {
            "dataset": "GSE12345",
            "groups": ["cancer", "healthy"]
        },
        "software": {
            "os": "Ubuntu 22.04",
            "python_version": "3.11.4",
            "r_version": "4.3.1",
            "packages": {
                "biopython": "1.81",
                "pandas": "2.0.3",
                "DESeq2": "1.40.0"
            }
        },
        "workflow": [
            {
                "step": 1,
                "name": "download_data",
                "tool": "sra-tools",
                "command": "prefetch SRR123456 SRR123457",
                "input": "SRR123456.sra",
                "output": "SRR123456.sra"
            },
            {
                "step": 2,
                "name": "quality_control",
                "tool": "fastqc",
                "command": "fastqc SRR123456.sra",
                "input": "SRR123456.sra",
                "output": "SRR123456_fastqc.html"
            },
            {
                "step": 3,
                "name": "alignment",
                "tool": "star",
                "command": "STAR --genomeDir /ref/hg38 --readFilesIn SRR123456.sra",
                "input": "SRR123456.sra",
                "output": "SRR123456_Aligned.bam"
            },
            {
                "step": 4,
                "name": "quantification",
                "tool": "featureCounts",
                "command": "featureCounts -a genes.gtf -o counts.txt SRR123456_Aligned.bam",
                "input": "SRR123456_Aligned.bam",
                "output": "counts.txt"
            },
            {
                "step": 5,
                "name": "differential_expression",
                "tool": "DESeq2",
                "command": "Rscript deseq2.R counts.txt",
                "input": "counts.txt",
                "output": "results.csv"
            }
        ],
        "results": {
            "total_genes": 20000,
            "significant_genes": 1523,
            "upregulated": 892,
            "downregulated": 631
        }
    }
    return record
```

{"step": 1, "name": "Quality control", "tool": "FastQC", "version": "0.11.9"},
{"step": 2, "name": "Trimming", "tool": "Trimmomatic", "version": "0.39"},
{"step": 3, "name": "Alignment", "tool": "STAR", "version": "2.7.10a"},
{"step": 4, "name": "Quantification", "tool": "featureCounts", "version": "2.0.3"},
{"step": 5, "name": "Normalization", "tool": "DESeq2", "version": "1.40.0"},
{"step": 6, "name": "Differential expression", "tool": "DESeq2", "version": "1.40.0"}
],
"parameters": {
"quality_threshold": 20,
"trimming": {"leading": 3, "trailing": 3},
"alignment": {"genome": "hg38"},
"differential_expression": {"alpha": 0.05, "lfcThreshold": 1}
},
"outputs": {
"significant_genes": 45,
"top_genes": ["BRCA1", "TP53", "ESR1", "MYC", "EGFR"],
"plots": ["pca_plot.png", "ma_plot.png", "volcano_plot.png", "heatmap.png"],
"report": "differential_expression_results.csv"
},
"limitations": [
"حجم العينة محدود (n=6)",
"تأثيرات دفعات محتملة (تم تضمينها في النموذج)",
"النتائج تحتاج إلى تحقق تجريبي",
"المرجع الجينومي (hg38) قد يختلف عن جينوم العينة الفعلي"

```
]

}
```

```
    return record
# حفظ السجل
import json
record = create_reproducibility_record()
with open("reproducibility_record.json", "w") as f:
    json.dump(record, f, indent=2)
print("تم حفظ سجل إعادة الإنتاج.")
```

في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. في سير العمل، يجب قراءة العلاقة بين المدخلات والمخرجات والموارد والسجل باعتبارها رسمًا سببيًا للتحليل. نجاح خطوة منفردة لا يثبت أن الخط كله قابل لإعادة الإنتاج؛ الإصدار والبيئة والبيانات المرجعية جزء من التعريف الكامل للعملية.

This example shows how to build a comprehensive record for an RNA-seq analysis project, covering the question, data, software, processing steps, parameters, outputs, and limitations.

6. Documentation as Part of the Workflow

هذا المثال يوضح كيف يمكن بناء سجل شامل لمشروع تحليل RNA-seq، يغطي السؤال، والبيانات، والبرمجيات، وخطوات المعالجة، والمعاملات، والنتائج، والقيود.

Documentation is not a final step added at the end of the project; it is part of the workflow itself. Documentation should begin before data collection, continue during analysis, and end after publication. Every decision, every change, and every exception should be recorded immediately, not weeks or months later.

6. التوثيق كجزء من سير العمل

التوثيق ليس خطوة نهائية تُضاف في نهاية المشروع؛ بل هو جزء من سير العمل نفسه. يجب أن يبدأ التوثيق قبل جمع البيانات، ويستمر أثناء التحليل، وينتهي بعد النشر. كل قرار، وكل تغيير، وكل استثناء يجب أن يُسجل فوراً، وليس بعد أسابيع أو شهور.

مبدأ: دوّن كل قرار فور اتخاذه. لا تعتمد على الذاكرة.

سجل الأوامر، وسجل المعاملات، وسجل الأخطاء، وسجل القرارات.

Language / نوع الكود: **Bash / Shell**

```
# مثال: سجل أوامر Bash في (command log)
# هذا سجل بالأوامر المستخدمة في المشروع
```

```
commands = """
# 2025-03-15: تحميل البيانات
fastq-dump --split-files SRR123456
# 2025-03-17: قص المحولات
trimmomatic PE SRR123456_1.fastq SRR123456_2.fastq \
output_1.fastq output_2.fastq \
ILLUMINACLIP:adapters.fa:2:30:10 LEADING:3 TRAILING:3
# 2025-03-16: فحص الجودة
fastqc SRR123456_1.fastq
# 2025-03-18: محاذاة
STAR --genomeDir hg38 --readFilesIn output_1.fastq output_2.fastq --outFileNamePrefix aligned_
"""
```

7. Limits of Documentation and Reproducibility

Even with the best documentation, reproducibility may not always be complete. Some reasons:

7. حدود التوثيق وإعادة الإنتاج

حتى مع أفضل التوثيق، قد لا تكون إعادة الإنتاج كاملة دائماً. بعض الأسباب:

· تغيرات نظام التشغيل: بعض الحزم تتصرف بشكل مختلف على أنظمة تشغيل مختلفة.

· Operating system changes: some packages behave differently on different operating systems.

· تغيرات الأجهزة: قد تختلف النتائج بين معالجات مختلفة بسبب اختلافات في دقة الحساب.

· Hardware changes: results may differ between different processors due to differences in calculation precision.

· البيانات الديناميكية: بعض قواعد البيانات تتغير باستمرار، حتى مع توثيق الإصدار (قد لا يكون الإصدار السابق متاحاً).

· Dynamic data: some databases change constantly, even with version documentation (the previous version may not be available).

· القرارات البشرية: بعض القرارات غير قابلة للتوثيق الكامل (مثل سبب اختيار عتبة معينة بناءً على فحص بصري).

· Human decisions: some decisions are not fully documentable (e.g., why a specific threshold was chosen based on visual inspection).

مبدأ: استهدف إعادة الإنتاج قدر الإمكان، واعترف بالقيود.

التوثيق الجيد يقلل من عدم اليقين، لكنه لا يزيله تماماً.

8. From Documentation to Good Practice

Good documentation practice includes:

8. من التوثيق إلى الممارسة الجيدة

الممارسة الجيدة في التوثيق تتضمن:

· استخدام أنظمة التحكم في الإصدارات (مثل Git) لتتبع تغييرات الكود.

· استخدام البيئات الافتراضية (مثل venv، conda) لتجميد الاعتماديات.

· استخدام دفاتر الملاحظات التفاعلية (مثل Jupyter Notebook) لتوثيق التحليل خطوة بخطوة.

· Using version control systems (such as Git) to track code changes.

· كتابة سجلات الأوامر (command logs) لتسجيل كل أمر تم تشغيله.

· Using virtual environments (such as conda, venv) to freeze dependencies.

· تسجيل المعاملات والإصدارات في ملفات منفصلة (مثل environment.yml، requirements.txt).

· Using interactive notebooks (such as Jupyter Notebook) to document analysis step by step.

· مشاركة البيانات والكود والتوثيق مع المنشور.

· Writing command logs to record every command run.

· Recording parameters and versions in separate files (such as requirements.txt, environment.yml).

· Sharing data, code, and documentation with the publication.

القاعدة الذهبية: إذا لم تُوثقه، لم يحدث.

Chapter Summary

خلاصة الفصل

Documentation and reproducibility are the foundation of the scientific method in the digital age. The reproducibility record should cover the research question, data sources, software environment and versions, processing steps, parameters, outputs, and limitations. Documenting limitations is not a flaw, but part of the scientific result. Changes in sources and versions are information to be interpreted, not errors to be hidden. Documentation should be part of the workflow, not a final step. Even with the best documentation, reproducibility may

not be complete, but good documentation reduces uncertainty. Good practice includes using Git, virtual environments, notebooks, and command logs. The next chapter will move to Part 14 — The Integrated Capstone Project.

التوثيق وقابلية إعادة الإنتاج هما أساس المنهج العلمي في العصر الرقمي. سجل إعادة الإنتاج يجب أن يغطي السؤال العلمي، ومصادر البيانات، والبيئة البرمجية والإصدارات، وخطوات المعالجة، والمعاملات، والنتائج، والقيود. توثيق القيود ليس عيباً، بل جزء من النتيجة العلمية. التغيرات في المصادر والإصدارات هي معلومات يجب تفسيرها، وليس أخطاء يجب إخفاؤها. التوثيق يجب أن يكون جزءاً من سير العمل، وليس خطوة نهائية. حتى مع أفضل التوثيق، قد لا تكون إعادة الإنتاج كاملة، لكن التوثيق الجيد يقلل من عدم اليقين. الممارسة الجيدة تشمل استخدام Git، والبيئات الافتراضية، ودفاتر الملاحظات، وسجلات الأوامر. الفصل القادم سينتقل إلى الباب 14 — مشروع التخرج المتكامل.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "Documentation and Reproducibility" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "التوثيق وقابلية إعادة الإنتاج" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "good documentation guarantees complete reproducibility." This is incorrect; good documentation increases reproducibility, but does not guarantee it due to hardware, operating system, and dynamic data changes. The correct statement is: "good documentation increases reproducibility and reduces uncertainty, but does not guarantee complete reproducibility due to factors outside documentation." A simple test: Practical check: "Can someone else fully rerun my analysis and obtain the same results?".

مثال على التجاوز: القول إن "التوثيق الجيد يضمن إعادة الإنتاج الكاملة". هذا غير صحيح؛ التوثيق الجيد يزيد من قابلية إعادة الإنتاج، لكنه لا يضمنها بسبب تغيرات الأجهزة وأنظمة التشغيل والبيانات الديناميكية. الصحيح هو: "التوثيق الجيد يزيد من قابلية إعادة الإنتاج ويقلل من عدم اليقين، لكنه لا يضمن إعادة إنتاج كاملة بسبب عوامل خارج نطاق التوثيق." اختبار بسيط: فحص تطبيقي: "هل يمكن لشخص آخر إعادة تشغيل تحليلي بالكامل والحصول على نفس النتائج؟".

Example: A reproducibility record documenting analysis steps is a computational representation of the process. Claiming that "these results prove a biological fact" is a biological claim not proven by documentation alone. Documentation shows how results were produced, but proving a biological fact requires independent evidence. Practical test: before inferring a biological fact from a documented analysis, Practical check: "Is the result supported by independent evidence?".

مثال: سجل إعادة الإنتاج يوثق خطوات التحليل هو تمثيل حسابي للعملية. الادعاء بأن "هذه النتائج تثبت حقيقة بيولوجية" هو ادعاء بيولوجي لا يثبته التوثيق وحده. التوثيق يوضح كيف أُنتجت النتائج، لكن إثبات الحقيقة البيولوجية يحتاج إلى أدلة مستقلة. اختبار عملي: قبل أن تستنتج حقيقة بيولوجية من تحليل موثق، فحص تطبيقي: "هل النتيجة مدعومة بأدلة مستقلة؟"ـ

مثال تطبيقي على سجل إعادة الإنتاج:

· المدخلات: البيانات الخام، وصف المشروع، السؤال العلميـ

· الافتراضات: أن البيانات تمثل العينة بدقة، وأن البرمجيات تعمل كما هو متوقعـ

· المخرجات: سجل شامل يوثق كل جوانب التحليلـ

· القيود: السجل يعتمد على دقة التوثيق، وقد لا يغطي جميع القرارات البشرية أو العوامل الخارجيةـ

Applied example of a reproducibility record:

اختبار: قبل اعتماد سجل إعادة الإنتاج، تحقق من اكتماله ودقة توثيقهـ

· Inputs: raw data, project description, research question.

· Assumptions: that the data accurately represents the sample, and that the software works as expected.

· Outputs: a comprehensive record documenting all aspects of the analysis.

· Limitations: the record depends on documentation accuracy, and may not cover all human decisions or external factors.

Test: before relying on a reproducibility record, verify its completeness and documentation accuracy.

يجب تسجيل:

· المصدر: البيانات والمجلات والمراجع المستخدمةـ

· الإصدار: إصدارات البرمجيات وقواعد البياناتـ

· المعلمات: جميع المعاملات والعتبات المستخدمةـ

· التحويلات: خطوات المعالجة بالترتيبـ

اختبار: فحص تطبيقي: "لو أعطيت هذا السجل لزميل بعد عام، هل يستطيع فهم ما فعلته وإعادة تشغيل التحليل دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍـ

Record:

· Provenance: data, journals, and references used.

· Version: software and database versions.

· Parameters: all parameters and thresholds used.

· Transformations: processing steps in order.

Test: Practical check: "If I gave this record to a colleague a year later, could they understand what I did and rerun the analysis without my help?" If the answer is no, the documentation is insufficient.

قبل الاعتماد على سجل إعادة الإنتاج في سياق حقيقي، اختبر عملية التوثيق على مشروع صغير باستخدام بيانات اصطناعية. أنشئ مشروعاً وهمياً، وطبق نفس عملية التوثيق، ثم حاول إعادة تشغيله من التوثيق فقط. إذا نجحت، فأنت واثق من أن عملية التوثيق كافية. إذا فشلت، فأنت تعرف أين تحتاج إلى تحسين.

Before relying on a reproducibility record in a real context, test the documentation process on a small project using synthetic data. Create a dummy project, apply the same documentation process, then try to rerun it from the documentation alone. If you succeed, the workflow is confident the documentation process is sufficient. If you fail, you know where the analysis requires to improve.

Part 14 — Integrated Capstone Project

الباب 14 — مشروع التخرج المتكامل

الفكرة المركزية: دمج R، Bash، Biopython، Python، الأمن، الاختبار.

Chapter 55 — The Integrated Brick by Brick Capstone

الفصل 55 — مشروع Brick by Brick

شرح أعمق للمفهوم

Deeper Conceptual Explanation

المشروع المتكامل هو تمرين على تكامل الأدوات والطبقات: بيانات اصطناعية، Python و Bash، Biopython، R، فحوص دفاعية، سجلات، ورسوم وتقرير. لا ينبغي أن تُقرأ مخرجاته كاكتشافات بيولوجية جديدة؛ قيمته في إظهار كيف يمكن بناء سير عمل قابل للتتبع من البيانات إلى التقرير مع حدود واضحة.

The integrated project is an exercise in connecting layers: synthetic data, Python and Biopython, Bash, R, defensive checks, logs, visualizations, and a report. Its outputs should not be read as new biological discoveries; its value is showing how a traceable workflow can move from data to report while keeping its limits explicit.

شرح أعمق للمفهوم

Deeper Conceptual Explanation

We have now reached the most important stop on the journey: the integrated project. After learning how to set up the environment, understand bioinformatics, read files, analyze them, test hypotheses, and document the project's work, it is time to bring all these skills together in one integrated project. This chapter is the culmination of everything the preceding sections have established in this book. The final project brings together the book's stages without turning it into a sensitive biological experiment. The data starts synthetically, then passes through organized preparation, sequential analysis, neutral defensive checks, statistical and visual summarization, and ends with a report explaining every decision. The educational value is seeing how the layers connect: data quality affects analysis, analysis affects statistics, and statistics affects interpretation. This project gives you the opportunity to apply everything the preceding sections have established in one coherent context, with a focus on traceability and scientific reporting rather than sensitive laboratory experiments.

1. Project Overview

وصلنا الآن إلى محطة الرحلة الأهم: المشروع المتكامل. بعد أن تعلمت كيف تنشئ البيئة، وتفهم المعلوماتية الحيوية، وتقرأ الملفات، وتحللها، وتختبر الفرضيات، وتوثق عملك، حان الوقت لتجمع كل هذه المهارات في مشروع واحد متكامل. هذا الفصل هو تتويج لجميع ما تعلمته في هذا الكتاب. المشروع النهائي يجمع مراحل الكتاب دون تحويلها إلى تجربة بيولوجية حساسة. تبدأ البيانات اصطناعياً، ثم تمر بتجهيز منظم، وتحليل تسلسلي، وفحوص دفاعية محايدة، ثم تلخيص إحصائي وتصويري، وتنتهي بتقرير يوضح كل قرار. القيمة التعليمية هي رؤية كيف تتصل الطبقات: جودة البيانات تؤثر في التحليل، والتحليل يؤثر في الإحصاء، والإحصاء يؤثر في التفسير. هذا المشروع يمنحك الفرصة لتطبيق كل ما تعلمته في سياق واحد متماسك، مع التركيز على قابلية التتبع والتوثيق العلمي بدلاً من التجارب المخبرية الحساسة.

The "Brick by Brick" integrated project is an educational application that brings together the tools and concepts you learned in this book. The goal is not to produce a new biological discovery, but to demonstrate how different tools interconnect in a typical bioinformatics workflow. The project goes through the following stages:

1. نظرة عامة على المشروع

المشروع المتكامل "Brick by Brick" هو تطبيق تعليمي يجمع بين الأدوات والمفاهيم التي تعلمتها في هذا الكتاب. الهدف ليس إنتاج اكتشاف بيولوجي جديد، بل إظهار كيف تترابط الأدوات المختلفة في سير عمل معلوماتي حيوي نموذجي. المشروع يمر بالمراحل التالية:

المرحلة 1: تحضير البيانات الاصطناعية (Synthetic Data Preparation).

نقوم بإنشاء بيانات تسلسل اصطناعية تحاكي بيانات حقيقية، لكنها آمنة وقابلة للتكرار. تشمل تسلسلات DNA، وبيانات وصفية، وبيانات تعبير جيني محاكاة.

المرحلة 2: إدارة سير العمل باستخدام Bash (Bash Workflow Management).

المرحلة 3: تحليل البيانات باستخدام Python و Biopython (Python/Biopython Analysis).

نستخدم Bash لتنظيم الملفات، وتشغيل الأدوات، وإدارة سير العمل بطريقة منظمة وقابلة للتتبع.

نقرأ الملفات، ونحلل التسلسلات، ونحسب الإحصائيات الأساسية، ونجري عمليات بيولوجية (مثل التكامل العكسي والترجمة).

المرحلة 4: فحوصات أمنية دفاعية (Defensive Security Checks).

نطبق فحوصات أمنية بسيطة على البيانات، مثل التحقق من صحة التسلسلات، والكشف عن الأنماط المشبوهة، وتطبيق مبدأ fail-closed.

المرحلة 5: التحليل الإحصائي والتصور باستخدام R (R-based Statistics and Visualization).

نستخدم R لتحليل بيانات التعبير، وإجراء اختبارات إحصائية، وإنشاء رسوم بيانية تلخص النتائج.

2. Stage 1: Synthetic Data Preparation

المرحلة 6: التوثيق والتقرير النهائي (Documentation and Final Report).

Before we start any analysis, we need data. In this project, we will use completely synthetic data. This ensures the project is repeatable, safe, and does not depend on internet connectivity. We will create three types of data:

نوثق كل خطوة، وننشئ تقريراً علمياً يوضح السؤال، والمنهج، والنتائج، والقيود.

2. المرحلة 1: تحضير البيانات الاصطناعية

قبل أن نبدأ أي تحليل، نحتاج إلى بيانات. في هذا المشروع، سنستخدم بيانات اصطناعية بالكامل. هذا يضمن أن المشروع قابل للتكرار، وآمن، ولا يعتمد على اتصال بالإنترنت. سننشئ ثلاثة أنواع من البيانات:

أولاً: تسلسلات DNA اصطناعية (FASTA).

سننشئ 20 تسلسلاً DNA بطول 100-300 قاعدة، مع هويات وصفية. بعض التسلسلات ستكون عشوائية، وبعضها سيحتوي على أنماط معروفة (مثل مواقع بدء الترجمة، أو مواقع ارتباط).

ثانياً: بيانات تعبير جيني محاكاة (Counts).

سنحاكي بيانات RNA-seq: 100 جين، و12 عينة (6 سليمة، 6 مريضة). بعض الجينات ستكون مختلفة بين المجموعتين.

ثالثاً: بيانات وصفية (Metadata).

Synthetic Data Preparation Code (Python).

سننشئ جدولاً يربط كل عينة بحالتها (سليم/مريض)، ومجموعة الدفعة، ومعلومات إضافية.

كود تحضير البيانات الاصطناعية (Python).

import os
import numpy as np
import pandas as pd
import random
from Bio.Seq import Seq
from Bio import SeqIO
from Bio.SeqRecord import SeqRecord
# ==
# brick_brick_project/data_preparation.py
# ==
# توليد البيانات الاصطناعية للمشروع المتكامل
# Synthetic data generation for the integrated project
def generate_synthetic_sequences(n_sequences=20, min_len=100,
max_len=300, output_file="data/synthetic_sequences.fasta"):
"""

توليد تسلسلات DNA اصطناعية مع هويات وصفية.

Generate synthetic DNA sequences with identifiers and descriptions.

```
    """
    os.makedirs("data", exist_ok=True)
# ============================================================
# 1. توليد تسلسلات DNA اصطناعية (FASTA)
# 1. Generate synthetic DNA sequences (FASTA)
# ============================================================
records = []
    for i in range(n_sequences):
        length = random.randint(min_len, max_len)
```

```
    seq = ''.join(random.choices(bases, k=length))
bases = ['A', 'T', 'C', 'G']
  # أنماط معروفة لإضافتها إلى بعض التسلسلات
  known_patterns = ["ATGGCC", "TAG", "GCTAGCTAGC", "GGGCCC", "AATTCCGG"]
# إضافة نمط معروف إلى بعض التسلسلات
    if i < 5:
      pattern = random.choice(known_patterns)
      pos = random.randint(0, length - len(pattern))
      seq = seq[:pos] + pattern + seq[pos+len(pattern):]
seq_obj = Seq(seq)
    record = SeqRecord(
```

seq_obj,

```
      id=f"SYN_{i+1:03d}",
      description=f"Synthetic sequence {i+1} | length={length} | {'with_pattern' if i < 5 else
'random'}"
```

)

```
    records.append(record)
SeqIO.write(records, output_file, "fasta")
  print(f"[✓] تم توليد {n_sequences} تسلسلاً في {output_file}")
```

return records

==

2. توليد بيانات تعبير جيني محاكاة (Counts)

2. Generate simulated gene expression data (Counts)

==

def generate_expression_data(n_genes=100, n_samples=12, n_diseased=6, output_prefix="data/expression"):

"""

توليد بيانات تعبير جيني محاكاة: 100 جين، 12 عينة، 6 سليمة، 6 مريضة.

Generate simulated gene expression data: 100 genes, 12 samples, 6 healthy, 6 diseased.

```
  """
  os.makedirs("data", exist_ok=True)
np.random.seed(42)
```

n_healthy = n_samples - n_diseased

توليد أعداد قراءات أساسية من توزيع بواسون

base_counts = np.random.poisson(lam=50, size=(n_genes, n_samples))

for gene_idx in diff_genes:

زيادة التعبير في العينات المريضة

base_counts[gene_idx, n_healthy:] = base_counts[gene_idx, n_healthy:] * np.random.uniform(1.5, 3.0)

```
# جعل بعض الجينات مختلفة بين المجموعتين
    n_diff_genes = 20
    diff_genes = np.random.choice(n_genes, n_diff_genes, replace=False)
# إضافة بعض الضوضاء العشوائية
    noise = np.random.normal(0, 2, (n_genes, n_samples))
    counts = np.maximum(0, base_counts + noise).astype(int)
# إنشاء DataFrame
    gene_names = [f"GENE_{i+1:03d}" for i in range(n_genes)]
    sample_names = [f"S_{i+1:03d}" for i in range(n_samples)]
    df_counts = pd.DataFrame(counts, index=gene_names, columns=sample_names)
# حفظ بيانات الأعداد
    counts_file = f"{output_prefix}_counts.csv"
    df_counts.to_csv(counts_file)
print(f"[✓] تم توليد بيانات التعبير ({n_genes} جين × {n_samples} عينة) في {counts_file}")

conditions = ["healthy"] * n_healthy + ["diseased"] * n_diseased
    batches = [random.choice(["batch_A", "batch_B"]) for _ in range(n_samples)]
# ============================================================
    # 3. توليد البيانات الوصفية (Metadata)
    # 3. Generate metadata
    # ============================================================
metadata = pd.DataFrame({
        "sample": sample_names,
        "condition": conditions,
        "batch": batches,
        "age": np.random.randint(25, 65, n_samples),
        "sex": np.random.choice(["M", "F"], n_samples)
    })
#بعض العينات في دفعة مختلفة لخلق تأثير دفعة محاكى
    for i in range(2, 4):
        batches[i] = "batch_C"
    for i in range(8, 10):
        batches[i] = "batch_C"
metadata_file = f"{output_prefix}_metadata.csv"
    metadata.to_csv(metadata_file, index=False)
    print(f"[✓] تم توليد البيانات الوصفية في {metadata_file}")
return df_counts, metadata
# ============================================================
# 4. توليد ملفات FASTA اصطناعية إضافية للاختبارات
# 4. Generate additional synthetic FASTA files for testing
# ============================================================
# تسلسل آمن
safe_seq = Seq("ATGCGTACGTAGCTAGCTAGCTAGCTAGC")
```

safe_record = SeqRecord(safe_seq, id="SAFE_001", description="Safe sequence for testing")

def generate_test_sequences():

"""توليد ملفات FASTA إضافية لاختبار الفحوصات الدفاعية."""

```
    os.makedirs("data", exist_ok=True)
# تسلسل يحتوي على نمط مشبوه (محاكاة تعليمية)
    suspicious_pattern = "GACTAGCTAGCTAGCTAGCNGG"
    suspicious_seq = Seq(suspicious_pattern)
    suspicious_record = SeqRecord(suspicious_seq, id="SUS_001", description="Suspicious pattern (educational)")
# تسلسل آخر آمن
    safe_seq2 = Seq("GCTAGCTAGCTAGCTAGCTAGCTAGCTA")
    safe_record2 = SeqRecord(safe_seq2, id="SAFE_002", description="Another safe sequence")
records = [safe_record, suspicious_record, safe_record2]
    output_file = "data/test_sequences.fasta"
    SeqIO.write(records, output_file, "fasta")
    print(f"[✓] تم توليد ملف اختبار {output_file}")
```

return records

==

5. تشغيل جميع وظائف التوليد

5. Run all generation functions

==

توليد التسلسلات

sequences = generate_synthetic_sequences(n_sequences=20)

if __name__ == "__main__":

```
    print("=" * 60)
```

print("بدء توليد البيانات الاصطناعية للمشروع المتكامل")

```
    print("Starting synthetic data generation for the integrated project")
    print("=" * 60)
# توليد بيانات التعبير
```

counts_df, metadata_df = generate_expression_data(n_genes=100, n_samples=12)

```
# توليد ملفات الاختبار
    test_records = generate_test_sequences()
print("\n" + "=" * 60)
```

print("اكتمل توليد البيانات الاصطناعية!")

```
    print("Synthetic data generation complete!")
    print("=" * 60)
    print("\nملفات تم إنشاؤها:")
    print("  - data/synthetic_sequences.fasta (20 تسلسلاً DNA)")
    print("  - data/expression_counts.csv (100 عينة × 12 جين)")
    print("  - data/expression_metadata.csv (بيانات وصفية للعينات)")
    print("  - data/test_sequences.fasta (ملف اختبار)")
```

3. Stage 2: Bash Workflow Management

After generating the data, we need to organize the workflow. Bash is the ideal tool for managing files, running scripts, and connecting different stages. We will create a main Bash script that executes everything in an organized manner.

3. المرحلة 2: إدارة سير العمل باستخدام Bash

بعد توليد البيانات، نحتاج إلى تنظيم سير العمل. Bash هو الأداة المثالية لإدارة الملفات، وتشغيل السكربتات، وربط المراحل المختلفة. سننشئ سكربت Bash رئيسياً ينفذ كل شيء بطريقة منظمة.

Language / نوع الكود: **Bash / Shell**

```
# إعداد المتغيرات
PROJECT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
DATA_DIR="$PROJECT_DIR/data"
RESULTS_DIR="$PROJECT_DIR/results"
LOGS_DIR="$PROJECT_DIR/logs"
SCRIPTS_DIR="$PROJECT_DIR/scripts"
#!/bin/bash
# ============================================================
# brick_brick_project/run_pipeline.sh
# سكربت Bash الرئيسي لإدارة سير عمل المشروع المتكامل
# Main Bash script for managing the integrated project workflow
# ============================================================
# إنشاء المجلدات
mkdir -p "$DATA_DIR" "$RESULTS_DIR" "$LOGS_DIR" "$SCRIPTS_DIR"
# تسجيل بدء التشغيل
```

```
echo "============================================================"
echo "بدء تشغيل مشروع Brick by Brick المتكامل"
echo "Starting Brick by Brick Integrated Project"
echo "التاريخ: $(date)"
echo ""
```

```
# ============================================================
# المرحلة 1: توليد البيانات الاصطناعية
# Stage 1: Generate synthetic data
# ============================================================
```

```
echo "[المرحلة 1] توليد البيانات الاصطناعية..."
echo "[Stage 1] Generating synthetic data..."
python3 "$SCRIPTS_DIR/data_preparation.py" 2>&1 | tee "$LOGS_DIR/01_data_generation.log"
if [ ${PIPESTATUS[0]} -ne 0 ]; then
    echo "❌ فشل توليد البيانات!" >&2
    exit 1
```

```
fi

echo "✅ اكتمل توليد البيانات"

echo ""

# ============================================================
# المرحلة 2: تحليل التسلسلات باستخدام Biopython
# Stage 2: Sequence analysis using Biopython
# ============================================================

echo "[المرحلة 2] تحليل التسلسلات باستخدام Biopython..."

echo "[Stage 2] Sequence analysis using Biopython..."

python3 "$SCRIPTS_DIR/sequence_analysis.py" \
    --input "$DATA_DIR/synthetic_sequences.fasta" \
    --output "$RESULTS_DIR/sequence_stats.csv" \
    2>&1 | tee "$LOGS_DIR/02_sequence_analysis.log"

if [ ${PIPESTATUS[0]} -ne 0 ]; then
    echo "❌ فشل تحليل التسلسلات!" >&2
    exit 1
fi

echo "✅ اكتمل تحليل التسلسلات"

echo ""

# ============================================================
# المرحلة 3: الفحوصات الأمنية الدفاعية
# Stage 3: Defensive security checks
# ============================================================

echo "[المرحلة 3] تشغيل الفحوصات الأمنية الدفاعية..."

echo "[Stage 3] Running defensive security checks..."

python3 "$SCRIPTS_DIR/security_checks.py" \
    --input "$DATA_DIR/test_sequences.fasta" \
    --output "$RESULTS_DIR/security_report.csv" \
    2>&1 | tee "$LOGS_DIR/03_security_checks.log"

if [ ${PIPESTATUS[0]} -ne 0 ]; then
    echo "❌ فشل الفحوصات الأمنية!" >&2
    exit 1
fi
```

```
echo "✅ اكتملت الفحوصات الأمنية"

echo ""

# ============================================================
# المرحلة 4: تحليل التعبير باستخدام R
# Stage 4: Expression analysis using R
# ============================================================

echo "[المرحلة 4] تحليل التعبير والإحصاء باستخدام R..."

echo "[Stage 4] Expression analysis and statistics using R..."

Rscript "$SCRIPTS_DIR/expression_analysis.R" \
    --counts "$DATA_DIR/expression_counts.csv" \
    --metadata "$DATA_DIR/expression_metadata.csv" \
    --output "$RESULTS_DIR" \
    2>&1 | tee "$LOGS_DIR/04_expression_analysis.log"

if [ ${PIPESTATUS[0]} -ne 0 ]; then
    echo "❌ فشل تحليل التعبير!" >&2
    exit 1
fi

echo "✅ اكتمل تحليل التعبير"

echo ""

# ============================================================
# المرحلة 5: توليد التقرير النهائي
# Stage 5: Generate final report
# ============================================================

echo "[المرحلة 5] توليد التقرير النهائي..."

echo "[Stage 5] Generating final report..."

python3 "$SCRIPTS_DIR/generate_report.py" \
    --results "$RESULTS_DIR" \
    --output "$RESULTS_DIR/final_report.md" \
    2>&1 | tee "$LOGS_DIR/05_report_generation.log"

if [ ${PIPESTATUS[0]} -ne 0 ]; then
    echo "❌ فشل توليد التقرير!" >&2
    exit 1
fi
```

```
"✅ اكتمل التقرير النهائي" echo

echo ""
```

```
# ============================================================
# عرض الملخص النهائي
# Final summary
# ============================================================
```

```
echo ""

"✅ اكتمل المشروع بنجاح!" echo

echo "✅ Project completed successfully!"

echo ""

echo "النتائج متاحة في: $RESULTS_DIR"

"السجلات متاحة في: $LOGS_DIR" echo

echo ""

"ملفات النتائج الرئيسية:" echo

ls -la "$RESULTS_DIR"

echo ""

"انتهى التشغيل في: $(date)" echo
```

في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. في سير العمل، يجب قراءة العلاقة بين المدخلات والمخرجات والموارد والسجل باعتبارها رسمًا سببيًا للتحليل. نجاح خطوة منفردة لا يثبت أن الخط كله قابل لإعادة الإنتاج؛ الإصدار والبيئة والبيانات المرجعية جزء من التعريف الكامل للعملية. أوامر الطرفية حساسة للسياق: نظام التشغيل، PATH، مجلد العمل، صلاحيات الملفات، وإصدارات الأدوات. لذلك يجب تسجيل الأمر كما نُفذ، مع مدخلاته ومخرجاته، بدل حفظ النتيجة وحدها.

4. Stage 3: Sequence Analysis Using Biopython

This stage reads the FASTA file, analyzes each sequence: calculates length, GC content, complement, reverse complement, translation, and detects known patterns.

4. المرحلة 3: تحليل التسلسلات باستخدام Biopython

هذه المرحلة تقرأ ملف FASTA، وتحلل كل تسلسل: تحسب الطول، ونسبة GC، والمتمم، والمتمم العكسي، والترجمة، وتكتشف الأنماط المعروفة.

```
#!/usr/bin/env python3
# ============================================================
# brick_brick_project/scripts/sequence_analysis.py
# تحليل التسلسلات باستخدام Biopython
# Sequence analysis using Biopython
# ============================================================
import os
import sys
import argparse
import pandas as pd
from Bio import SeqIO
from Bio.SeqUtils import gc_fraction
from Bio.SeqUtils import MeltingTemp as mt
def analyze_sequence(record):
    """
```

تحليل تسلسل واحد وحساب خصائصه.

Analyze a single sequence and calculate its properties.

```
    """
    seq = record.seq
    seq_str = str(seq)
    length = len(seq)
# خصائص أساسية
    gc_content = gc_fraction(seq)*100
# العمليات البيولوجية
    complement = seq.complement()
    reverse_complement = seq.reverse_complement()
# الترجمة (فقط إذا كان الطول مضاعفاً لـ 3)
    if length % 3 == 0 and set(seq_str.upper()).issubset(set("ATCG")):
        translation = seq.translate()
        has_stop = "*" in str(translation)
    else:
        translation = "N/A"
        has_stop = None
# اكتشاف الأنماط المعروفة
    known_patterns = {
```

"start_codon": "ATG",

"stop_codon_TAA": "TAA",

"stop_codon_TAG": "TAG",

"stop_codon_TGA": "TGA",

"promoter_TATA": "TATAAA",

"GC_rich": "GCGCGC"

}

```
pattern_matches = {}
    for name, pattern in known_patterns.items():
        if pattern in seq_str:
```

pattern_matches[name] = True

```
        else:
```

pattern_matches[name] = False

```
# درجة حرارة الانصهار التقريبية (للـ DNA)
    try:
        if length > 10:
            tm = mt.Tm_NN(seq, nn_table=mt.DNA_NN4)
        else:
            tm = mt.Tm_Wallace(seq)
    except:
        tm = None
result = {
```

"id": record.id,

"description": record.description,

"length": length,

"gc_content": gc_content,

"complement": str(complement),

"reverse_complement": str(reverse_complement),

"translation": str(translation) if translation != "N/A" else "N/A",

"has_stop_codon": has_stop,

**pattern_matches,

"tm_estimate": tm

}

```
return result
def main():
    parser = argparse.ArgumentParser(description="تحليل التسلسلات باستخدام Biopython")
    parser.add_argument("--input", required=True, help="ملف FASTA المدخل")
    parser.add_argument("--output", required=True, help="ملف CSV للمخرجات")
    args = parser.parse_args()
print(f"قراءة الملف: {args.input}")
# تحليل كل تسلسل
    results = []
    for i, record in enumerate(records):
        print(f"  تحليل التسلسل {i+1}: {record.id}")
        try:
            result = analyze_sequence(record)
            results.append(result)
        except Exception as e:
            print(f"  ⚠️ خطأ في تحليل {record.id}: {e}")
```

continue

```
# قراءة التسلسلات
    records = list(SeqIO.parse(args.input, "fasta"))
    print(f"تم قراءة {len(records)} تسلسلاً")
# حفظ النتائج كـ CSV
    df = pd.DataFrame(results)
    df.to_csv(args.output, index=False)
print(f"\n✅ تم حفظ النتائج في: {args.output}")
    print(f"📊 تم تحليل {len(results)} تسلسلاً بنجاح")
    print(f"  - متوسط الطول: {df['length'].mean():.1f}")
    print(f"  - متوسط نسبة GC: {df['gc_content'].mean():.1f}%")
if __name__ == "__main__":
    main()
```

5. Stage 4: Defensive Security Checks

This stage applies security checks on sequences, implements the fail-closed principle, and classifies sequences by educational risk level.

5. المرحلة 4: الفحوصات الأمنية الدفاعية

هذه المرحلة تطبق فحوصات أمنية على التسلسلات، وتطبق مبدأ fail-closed، وتصنف التسلسلات حسب درجة الخطر التعليمية.

```
#!/usr/bin/env python3
# ============================================================
# brick_brick_project/scripts/security_checks.py
# الفحوصات الأمنية الدفاعية
# Defensive security checks
# ============================================================
import os
import sys
import argparse
import pandas as pd
from Bio import SeqIO
# قائمة الأنماط المشبوهة (تعليمية فقط، لا تمثل تهديدات حقيقية)
# List of suspicious patterns (educational only, not real threats)
SUSPICIOUS_PATTERNS = {
    "pattern_A": "GACTAGCTAGCTAGCTAGC",
    "pattern_B": "AATTCCGG",
    "pattern_C": "GCGCGCGC",
    "toxin_sim": "GACTAGCTAGCTAGCTAGCNGG"
}

class SecurityCheck:
    """نظام فحوصات أمنية تعليمي يطبق مبدأ fail-closed."""

    def __init__(self, suspicious_patterns=None, threshold=0.8):
        self.suspicious_patterns = suspicious_patterns or SUSPICIOUS_PATTERNS
        self.threshold = threshold
```

self.audit_log = []
self.review_queue = []
def check_sequence(self, record):
"""

فحص تسلسل وتصنيفه.

Check a sequence and classify it.

```
    """
    seq_str = str(record.seq).upper()
    seq_id = record.id
# فحص الأنماط المشبوهة
    found_patterns = []
    max_similarity = 0
for pattern_name, pattern in self.suspicious_patterns.items():
        if pattern in seq_str:
            found_patterns.append(pattern_name)
            # حساب تشابه بسيط (محاكاة)
            sim = len(pattern) / len(seq_str) if len(seq_str) > 0 else 0
            max_similarity = max(max_similarity, sim)
        elif len(pattern) <= len(seq_str):
            # تشابه جزئي (محاكاة)
            for i in range(len(seq_str) - len(pattern) + 1):
                matches = sum(1 for a, b in zip(seq_str[i:i+len(pattern)], pattern) if a == b)
                sim = matches / len(pattern)
                if sim >= self.threshold * 0.6:
                    found_patterns.append(f"{pattern_name}_partial")
```

max_similarity = max(max_similarity, sim)

تصنيف المخاطر (تعليمي) #

if found_patterns and max_similarity >= self.threshold:
risk_level = "high"
action = "human_review_required"
elif found_patterns:
risk_level = "medium"
action = "human_review_recommended"
else:
risk_level = "low"
action = "auto_approve"

مبدأ fail-closed: في حالة عدم اليقين، التوقف والمراجعة #

if risk_level in ["high", "medium"] and self.threshold > 0.6:

محاكاة fail-closed: يطلب مراجعة بشرية #

self.review_queue.append({

"id": seq_id,

```
                "risk_level": risk_level,
                "found_patterns": found_patterns,
                "action": action,
                "status": "pending_review"
            })

# تسجيل في سجل التدقيق
        self.audit_log.append({
                "id": seq_id,
                "risk_level": risk_level,
                "found_patterns": found_patterns,
                "max_similarity": max_similarity,
                "action": action,
                "timestamp": "2025-03-20T12:00:00"
            })

        return {
                "id": seq_id,
                "risk_level": risk_level,
                "found_patterns": found_patterns,
                "action": action,
                "review_needed": action != "auto_approve"
            }

    def human_review(self, seq_id, reviewer, decision, rationale):
        """محاكاة المراجعة البشرية."""
        for item in self.review_queue:
            if item["id"] == seq_id:
                item["reviewer"] = reviewer
                item["decision"] = decision
                item["rationale"] = rationale
                item["status"] = "reviewed"
                break

# تسجيل المراجعة
        self.audit_log.append({
                "id": seq_id,
                "action": "human_review_completed",
```

```
"reviewer": reviewer,
"decision": decision,
"rationale": rationale,
"timestamp": "2025-03-20T12:30:00"
})
```

```
def main():
    parser = argparse.ArgumentParser(description="الفحوصات الأمنية الدفاعية")
    args = parser.parse_args()
print(f"قراءة ملف الاختبار: {args.input}")
# قراءة التسلسلات
    records = list(SeqIO.parse(args.input, "fasta"))
    print(f"تم قراءة {len(records)} تسلسلاً")
for record in records:
        result = checker.check_sequence(record)
        results.append(result)
        if result["review_needed"]:
print(f"⚠️ تسلسل {result['id']} يحتاج إلى مراجعة بشرية (مخاطر:
{result['risk_level']})")
```

```
# تشغيل الفحوصات
    checker = SecurityCheck()
    results = []
# عرض قائمة المراجعة
    if checker.review_queue:
print(f"\n📋 قائمة انتظار المراجعة البشرية: {len(checker.review_queue)} تسلسل")
```

```
        for item in checker.review_queue:
            print(f"  - {item['id']}: {item['risk_level']} - {item['found_patterns']}")
# حفظ النتائج
    df = pd.DataFrame(results)
    df.to_csv(args.output, index=False)
# محاكاة المراجعة البشرية لبعض الحالات
        print("\n🔄 محاكاة المراجعة البشرية...")
        for item in checker.review_queue[:2]:
checker.human_review(
item["id"],
"reviewer_001",
"approved",
"تم التحقق: النمط المشبوه جزء من منطقة محفوظة، لا يشكل خطراً تعليمياً"
)
```

```
        print(f"  ✅ تمت مراجعة {item['id']}")
if __name__ == "__main__":
    main()
print(f"\n✅ تم حفظ نتائج الفحوصات في: {args.output}")
    print(f"📊 الملخص:")
```

```
print(f"  - إجمالي التسلسلات: {len(results)}")
print(f"  - تحتاج مراجعة: {sum(1 for r in results if r['review_needed'])}")
print(f"  - تمت المراجعة: {len(checker.review_queue)} (محاكاة)")
```

عندما يتعامل الكود مع ملفات أو جداول، تصبح هوية الملف وترميزه ومساره جزءًا من النتيجة القابلة لإعادة الإنتاج. القراءة الناجحة للملف لا تعني أن محتواه صالح بيولوجيًا؛ لذلك يجب فصل فحص البنية عن تفسير البيانات. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

يُقرأ هذا المثال على ثلاث طبقات: المدخلات، والتحويل الذي تنفذه الدوال أو الكائنات، ثم المخرجات أو الاختبارات. وجود الفئة SecurityCheck يعني أن البيانات والسلوك جُمِعا في كائنات يمكن اختبارها وإعادة استخدامها، بدل تكرار المنطق في كل جزء من خط التحليل.

6. Stage 5: Statistical Analysis and Visualization Using R

This stage uses R to analyze expression data, perform statistical tests, and create plots that summarize the results.

6. المرحلة 5: التحليل الإحصائي والتصور باستخدام R

هذه المرحلة تستخدم R لتحليل بيانات التعبير، وإجراء اختبارات إحصائية، وإنشاء رسوم بيانية تلخص النتائج.

```
#!/usr/bin/env Rscript
# ============================================================
# brick_brick_project/scripts/expression_analysis.R
# تحليل التعبير والإحصاء باستخدام R
# Expression analysis and statistics using R
# ============================================================
# تحميل المكتبات
library(ggplot2)
library(reshape2)
library(ggrepel)
library(tidyverse)
# قراءة معاملات سطر الأوامر
args <- commandArgs(trailingOnly = TRUE)
counts_file <- args[1]  # data/expression_counts.csv
metadata_file <- args[2]  # data/expression_metadata.csv
output_dir <- args[3]  # results/
cat("قراءة بيانات التعبير...\n")
counts <- read.csv(counts_file, row.names = 1)
metadata <- read.csv(metadata_file)
```

```
cat(paste("بيانات:", nrow(counts), "جين ×", ncol(counts), "عينة\n"))

# ================================================================

# 1. التطبيع البسيط (CPM)

# 1. Simple normalization (CPM)

# ================================================================

cat("تطبيع البيانات (CPM)...\n")

lib_sizes <- colSums(counts)

cpm <- sweep(counts, 2, lib_sizes / 1e6, "/")

# ================================================================
# 2. تحليل التعبير التفاضلي (محاكاة باستخدام اختبار t)
# 2. Differential expression analysis (simulated using t-test)
# ================================================================
cat("تحليل التعبير التفاضلي...\n")

healthy_samples <- metadata$sample[metadata$condition == "healthy"]

diseased_samples <- metadata$sample[metadata$condition == "diseased"]

# حصر العينات الموجودة في البيانات

healthy_cols <- intersect(healthy_samples, colnames(cpm))

diseased_cols <- intersect(diseased_samples, colnames(cpm))

results <- data.frame(

  gene = rownames(cpm),
  log2fc = NA,
  p_value = NA,
  p_adj = NA,
  mean_healthy = NA,
  mean_diseased = NA

)

for (i in 1:nrow(cpm)) {

  healthy_vals <- as.numeric(cpm[i, healthy_cols])

  diseased_vals <- as.numeric(cpm[i, diseased_cols])

  results$mean_healthy[i] <- mean(healthy_vals)

  results$mean_diseased[i] <- mean(diseased_vals)

  # Log2 Fold Change
  if (results$mean_healthy[i] > 0 && results$mean_diseased[i] > 0) {

    results$log2fc[i] <- log2(results$mean_diseased[i] /
                              results$mean_healthy[i])

  } else {

    results$log2fc[i] <- 0
```

```
}
# اختبار t
 if (length(healthy_vals) >= 2 && length(diseased_vals) >= 2) {
t_result <- t.test(healthy_vals, diseased_vals)
results$p_value[i] <- t_result$p.value
} else {
results$p_value[i] <- 1
}
}
# تصحيح FDR (Benjamini-Hochberg)
results$p_adj <- p.adjust(results$p_value, method = "BH")
#تحديد الجينات ذات التعبير التفاضلي
results$significant <- results$p_adj < 0.05 & abs(results$log2fc) > 1
cat(paste("تم اكتشاف ", sum(results$significant, na.rm = TRUE), " جيناً مختلفاً\n"))
# 3.1: Volcano Plot
volcano_plot <- ggplot(results, aes(x = log2fc, y = -log10(p_adj), color =
significant)) +
 geom_point(alpha = 0.6, size = 2) +
 scale_color_manual(values = c("grey", "red")) +
 geom_hline(yintercept = -log10(0.05), linetype = "dashed", alpha = 0.5) +
 geom_vline(xintercept = c(-1, 1), linetype = "dashed", alpha = 0.5) +
labs(
  title = "Volcano Plot - Differential Expression",
  x = "Log2 Fold Change",
  y = "-Log10 Adjusted P-value"
) +
 theme_minimal() +
 theme(legend.position = "bottom")
# ============================================================
# 3. الرسوم البيانية
# 3. Plots
# ============================================================
cat("إنشاء الرسوم البيانية...\n")
ggsave(paste0(output_dir, "/volcano_plot.png"), volcano_plot, width = 8, height = 6)
cat("  ✅ volcano_plot.png\n")
# 3.2: Heatmap (أعلى 20 جيناً مختلفاً)
top_genes <- results[order(results$p_adj), ][1:min(20, sum(!
is.na(results$p_adj))), ]
top_counts <- cpm[rownames(cpm) %in% top_genes$gene, ]
top_counts_long <- melt(top_counts)
colnames(top_counts_long) <- c("Gene", "Sample", "Expression")
```

```
heatmap_plot <- ggplot(top_counts_long, aes(x = Sample, y = Gene, fill = Expression)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = median(top_counts_long$Expression)) +
  labs(title = "Heatmap - Top 20 Differential Genes") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
ggsave(paste0(output_dir, "/heatmap.png"), heatmap_plot, width = 10, height = 8)
cat("  ✅ heatmap.png\n")
# 3.3: MA Plot
ma_plot <- ggplot(results, aes(x = (mean_healthy + mean_diseased) / 2, y = log2fc, color = significant)) +
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(values = c("grey", "red")) +
  geom_hline(yintercept = 0, linetype = "solid", alpha = 0.7) +
  geom_hline(yintercept = c(-1, 1), linetype = "dashed", alpha = 0.5) +
  scale_x_log10() +
  labs(
    title = "MA Plot",
    x = "Mean Expression (log scale)",
    y = "Log2 Fold Change"
  ) +
  theme_minimal() +
  theme(legend.position = "bottom")
ggsave(paste0(output_dir, "/ma_plot.png"), ma_plot, width = 8, height = 6)
cat("  ✅ ma_plot.png\n")
# 3.4: PCA Plot
cat("  إنشاء PCA plot...\n")
pca_data <- t(cpm)
pca <- prcomp(pca_data, center = TRUE, scale. = TRUE)
pca_df <- as.data.frame(pca$x)
pca_df$sample <- rownames(pca_df)
pca_df <- merge(pca_df, metadata, by = "sample")
pca_plot <- ggplot(pca_df, aes(x = PC1, y = PC2, color = condition, shape = batch)) +
  geom_point(size = 4, alpha = 0.8) +
  labs(
    title = "PCA - Sample Clustering",
    x = paste0("PC1 (", round(summary(pca)$importance[2, 1] * 100, 1), "%)"),
    y = paste0("PC2 (", round(summary(pca)$importance[2, 2] * 100, 1), "%)")
  ) +
```

```
  theme_minimal() +
  theme(legend.position = "bottom")
ggsave(paste0(output_dir, "/pca_plot.png"), pca_plot, width = 8, height = 6)
cat("  ✅ pca_plot.png\n")
# ============================================================
# 4. حفظ النتائج
# 4. Save results
# ============================================================
write.csv(results, paste0(output_dir, "/differential_expression_results.csv"), row.names = FALSE)
cat(paste0("✅ تم حفظ النتائج في: ", output_dir,
"/differential_expression_results.csv\n"))
# ============================================================
# 5. ملخص النتائج
# 5. Summary of results
# ============================================================
cat("\n==========================================\n")
cat("ملخص تحليل التعبير التفاضلي:\n")
cat("==========================================\n")
cat(paste("إجمالي الجينات:", nrow(results), "\n"))
cat(paste("جينات ذات تعبير تفاضلي:", sum(results$significant, na.rm = TRUE), "\n"))
cat(paste("  - مرتفعة في المرضى:", sum(results$log2fc > 1 & results$significant,
na.rm = TRUE), "\n"))
cat(paste("  - منخفضة في المرضى:", sum(results$log2fc < -1 & results$significant,
na.rm = TRUE), "\n"))
cat("\nأعلى 5 جينات متفاوتة:\n")
print(head(results[order(results$p_adj), c("gene", "log2fc", "p_adj")], 5))
cat("\n✅ اكتمل تحليل R\n")
```

7. Stage 6: Final Report

This stage generates a final scientific report summarizing the entire project.

7. المرحلة 6: التقرير النهائي

هذه المرحلة تولد تقريراً علمياً نهائياً يلخص المشروع بأكمله.

```
#!/usr/bin/env python3
# ============================================================
# brick_brick_project/scripts/generate_report.py
# توليد التقرير النهائي
# Generate final report
# ============================================================
import os
import sys
import argparse
```

```
import pandas as pd
from datetime import datetime
def generate_report(results_dir, output_file):
    """
```

توليد تقرير علمي نهائي من نتائج المشروع.

Generate a final scientific report from project results.

```
    """
    # قراءة النتائج المختلفة
    try:
        seq_stats = pd.read_csv(f"{results_dir}/sequence_stats.csv")
        security_report = pd.read_csv(f"{results_dir}/security_report.csv")
        de_results = pd.read_csv(f"{results_dir}/differential_expression_results.csv")
    except FileNotFoundError as e:
        print(f"⚠️ ملف غير موجود: {e}")
        print("سيتم توليد تقرير جزئي")
        de_results = None
        seq_stats = None
        security_report = None
# بناء التقرير
    report = []
    report.append("# تقرير مشروع Brick by Brick المتكامل\n")
    report.append(f"**تاريخ التقرير:** {datetime.now().strftime('%Y-%m-%d %H:%M')}\n")
    report.append("---\n")
# 1. المقدمة
    report.append("## 1. المقدمة\n")
```

report.append("هذا التقرير يلخص نتائج المشروع المتكامل 'Brick by Brick'، الذي يدمج

أدوات المعلوماتية الحيوية في سير عمل واحد.\n")

report.append("المشروع يستخدم بيانات اصطناعية بالكامل، ويهدف إلى إظهار كيفية ترابط الأدوات

المختلفة.\n")

```
    report.append("\n---\n")
# 2. السؤال العلمي
    report.append("## 2. السؤال العلمي\n")
```

report.append("**السؤال:** ما هي الجينات ذات التعبير التفاضلي بين العينات السليمة والمريضة

في بيانات RNA-seq المحاكاة؟\n")

report.append("**المجموعات:** 6 عينات سليمة، 6 عينات مريضة.\n")

```
                                report.append("**البيانات:** 100 جين محاكاة.\n")
                                report.append("\n---\n")
                                # 3. تحليل التسلسلات
                                report.append("## 3. تحليل التسلسلات\n")
                                if seq_stats is not None:
                                    report.append(f"- عدد التسلسلات المحللة: {len(seq_stats)}\n")
                                    report.append(f"- متوسط الطول: {seq_stats['length'].mean():.1f} قاعدة\n")
                                    report.append(f"- متوسط نسبة GC: {seq_stats['gc_content'].mean():.1f}%\n")
```

report.append(f"- عدد التسلسلات التي تحتوي على كودون بدء (ATG):

{seq_stats['start_codon'].sum()}\n")

report.append(f"- عدد التسلسلات التي تحتوي على كودون توقف:

{seq_stats['has_stop_codon'].sum() if 'has_stop_codon' in seq_stats else 'N/A'}
\n")

```
  else:
```

report.append("*(بيانات تحليل التسلسلات غير متوفرة)*\n")

```
  report.append("\n---\n")
# 4. الفحوصات الأمنية
  report.append("## 4. الفحوصات الأمنية الدفاعية\n")
  if security_report is not None:
    report.append(f"- عدد التسلسلات التي تم فحصها: {len(security_report)}\n")
```

report.append(f"- التسلسلات التي تحتاج إلى مراجعة:

{security_report['review_needed'].sum() if 'review_needed' in
security_report else 0}\n")

report.append("- تم تطبيق مبدأ fail-closed: الحالات غير الواضحة تُرسل للمراجعة

البشرية\n")

```
  else:
```

report.append("*(بيانات الفحوصات الأمنية غير متوفرة)*\n")

```
  report.append("\n---\n")
if len(sig_genes) > 0:
        up_genes = sig_genes[sig_genes.get('log2fc', 0) > 0]
        down_genes = sig_genes[sig_genes.get('log2fc', 0) < 0]
        report.append(f"- مرتفعة في المرضى: {len(up_genes)}\n")
        report.append(f"- منخفضة في المرضى: {len(down_genes)}\n")
# 5. تحليل التعبير التفاضلي
  report.append("## 5. تحليل التعبير التفاضلي\n")
  if de_results is not None and len(de_results) > 0:
    sig_genes = de_results[de_results.get('significant', False)]
    report.append(f"- عدد الجينات المحللة: {len(de_results)}\n")
```

report.append(f"- عدد الجينات ذات التعبير التفاضلي: {len(sig_genes)}\n")

```
# أفضل 5 جينات
        top_genes = sig_genes.sort_values('p_adj').head(5)
        report.append("\n**أعلى 5 جينات ذات تعبير تفاضلي:**\n")
        report.append("| الجين | Log2 Fold Change | P-Value معدل |\n")
        report.append("|------|----------------|----------|\n")
        for _, row in top_genes.iterrows():
          report.append(f"| {row['gene']} | {row['log2fc']:.2f} | {row['p_adj']:.4f} |\n")
  else:
```

report.append("*(بيانات التعبير التفاضلي غير متوفرة)*\n")

```
report.append("\n---\n")
# 6. القيود
report.append("## 6. القيود\n")
```

report.append("1. **البيانات الاصطناعية**: النتائج لا تعكس الواقع البيولوجي الحقيقي.\n")

report.append("2. **حجم العينة**: 12 عينة فقط (6 سليمة، 6 مريضة) محدودة من الناحية الإحصائية.\n")

report.append("3. **التطبيع البسيط**: تم استخدام CPM بدلاً من طرق تطبيع أكثر تقدماً.\n")

report.append("4. **الاختبار الإحصائي**: تم استخدام اختبار t بدلاً من نماذج أكثر تعقيداً.\n")

report.append("5. **الفحوصات الأمنية**: تعليمية فقط، ولا تمثل نظاماً أمنياً حقيقياً.\n")

```
report.append("\n---\n")
# 8. معلومات البيئة
```

report.append("## 8. معلومات البيئة وإعادة الإنتاج\n")

```
report.append("### البرمجيات والإصدارات\n")
report.append("- **Python**: 3.11+\n")
report.append("- **Biopython**: 1.81+\n")
report.append("- **R**: 4.3+\n")
report.append("- **المكتبات المستخدمة**: pandas, numpy, ggplot2, DESeq2 (محاكاة)\n")
report.append("\n### هيكل المشروع\n")
report.append("```\n")
report.append("brick_brick_project/\n")
report.append("├── data/                    # البيانات الاصطناعية\n")
report.append("├── results/                 # النتائج والرسوم\n")
report.append("├── logs/                    # سجلات التشغيل\n")
report.append("├── scripts/                 # سكربتات التحليل\n")
report.append("│   ├── data_preparation.py\n")
report.append("│   ├── sequence_analysis.py\n")
report.append("│   ├── security_checks.py\n")
report.append("│   ├── expression_analysis.R\n")
report.append("│   └── generate_report.py\n")
report.append("└── run_pipeline.sh          # سكربت التشغيل الرئيسي\n")
report.append("```\n")
report.append("\n### كيفية إعادة التشغيل\n")
report.append("```bash\n")
report.append("# 1. استنساخ المشروع\n")
report.append("git clone brick_brick_project\n")
report.append("cd brick_brick_project\n")
report.append("\n# 2. إنشاء البيئة الافتراضية\n")
report.append("python -m venv .venv\n")
report.append("source .venv/bin/activate\n")
report.append("pip install biopython pandas numpy\n")
report.append("\n# 3. تشغيل المشروع بالكامل\n")
report.append("./run_pipeline.sh\n")
report.append("```\n")
report.append("\n---\n")
# 7. الخاتمة
report.append("## 7. الخاتمة\n")
```

report.append("هذا المشروع أظهر كيف يمكن دمج الأدوات المختلفة في سير عمل معلوماتي حيوي متكامل.\n")

report.append("تم استخدام بيانات اصطناعية لإظهار قابلية إعادة الإنتاج والأمان.\n")

report.append("المشروع يوضح الترابط بين جودة البيانات، التحليل، والإحصاء.\n")

```
    report.append("\n---\n")
# 9. المراجع
    report.append("## 9. المراجع\n")
    report.append("1. Cock, P. J. A. et al. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics.\n")
    report.append("2. Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.\n")
    report.append("3. Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis.\n")
```

report.append("4. هذا المشروع هو تطبيق تعليمي، ويستخدم بيانات اصطناعية بالكامل.\n")

```
    report.append("\n---\n")
```

report.append(f"*تم إنشاء هذا التقرير تلقائياً في {datetime.now().strftime('%Y-%m-%d %H:%M')}*\n")

```
# كتابة التقرير
    with open(output_file, 'w', encoding='utf-8') as f:
        f.write(''.join(report))
print(f"✅ تم توليد التقرير النهائي: {output_file}")
    return output_file
def main():
    parser = argparse.ArgumentParser(description="توليد التقرير النهائي")
    parser.add_argument("--results", required=True, help="مجلد النتائج")
    parser.add_argument("--output", required=True, help="ملف التقرير النهائي")
    args = parser.parse_args()
generate_report(args.results, args.output)
if __name__ == "__main__":
    main()
```

8. Project Summary and Outputs

When running the full project, the following outputs will be produced:

8. ملخص المشروع ومخرجاته

عند تشغيل المشروع بالكامل، ستنتج المخرجات التالية:

· data/synthetic_sequences.fasta: 20 تسلسلاً DNA اصطناعياً

مخرجات البيانات (Data Outputs):

· data/expression_counts.csv: 100 جين × 12 عينة (بيانات تعبير)

· data/expression_metadata.csv: بيانات وصفية للعينات

· data/test_sequences.fasta: ملف اختبار للفحوصات الأمنية.

مخرجات التحليل (Analysis Outputs):

· results/sequence_stats.csv: إحصائيات التسلسلات.

· results/security_report.csv: تقرير الفحوصات الأمنية.

· results/differential_expression_results.csv: نتائج التعبير التفاضلي.

· results/volcano_plot.png: مخطط البركان.

· results/ma_plot.png: مخطط MA

مخرجات الرسوم البيانية (Plot Outputs):

· results/pca_plot.png: مخطط PCA.

· results/heatmap.png: خريطة حرارية لأعلى 20 جيناً.

مخرجات السجلات والتقرير (Logs and Report):

· logs/: سجلات تشغيل كل مرحلة

· results/final_report.md: التقرير العلمي النهائي.

9. مخرجات التعلم

1. Central idea and its limits — Explain the central idea of "The Integrated Brick by Brick Capstone" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "مشروع Brick by Brick" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "the Brick by Brick project produces new biological discoveries that can be published in a scientific journal." This is incorrect; the project is entirely educational, uses synthetic data, and the goal is to demonstrate tool integration and concepts, not to produce new biological knowledge. The correct statement is: "the Brick by Brick project is an integrated educational application that combines various bioinformatics tools (Python, Biopython, Bash, R) in one coherent workflow, using synthetic data, to illustrate how analysis stages connect from raw data to final report, with a focus on traceability and reproducibility." A simple test: Practical check: "Do the results I obtained from this project represent biological reality, or are they just an educational exercise on bioinformatics tools?".

مثال على التجاوز: القول إن "مشروع Brick by Brick ينتج اكتشافات بيولوجية جديدة يمكن نشرها في مجلة علمية". هذا غير صحيح؛ المشروع تعليمي بالكامل، ويستخدم بيانات اصطناعية، والهدف هو إظهار تكامل الأدوات والمفاهيم، وليس إنتاج معرفة بيولوجية جديدة. الصحيح هو: "مشروع Brick by Brick هو تطبيق تعليمي متكامل يجمع بين أدوات المعلوماتية الحيوية المختلفة (Python، Biopython، Bash، R) في سير عمل واحد متماسك، باستخدام بيانات اصطناعية، بهدف توضيح كيفية ترابط مراحل التحليل من البيانات الخام إلى التقرير النهائي، مع التركيز على قابلية التتبع وإعادة الإنتاج." اختبار بسيط: فحص تطبيقي: "هل النتائج التي حصلت عليها من هذا المشروع تمثل حقيقة بيولوجية، أم أنها مجرد تمرين تعليمي على أدوات المعلوماتية الحيوية؟"ـ

Example: The differential expression results from the project (e.g., a list of genes with changed expression) are a computational representation based on synthetic data and simulated algorithms. Claiming that "these genes are actually associated with the disease in living organisms" is a biological claim not proven by this project. The analysis shows that the code works and the steps are correct, but it says nothing about biological reality. Practical test: before inferring any biological relationship from the project results, Practical check: "Is this data real or synthetic? Is this analysis designed to answer a real biological question or to illustrate an analytical methodology?".

مثال: نتائج تحليل التعبير التفاضلي من المشروع (مثل قائمة الجينات ذات التغيير في التعبير) هي تمثيل حسابي يعتمد على بيانات اصطناعية وخوارزميات محاكاة. الادعاء بأن "هذه الجينات مرتبطة فعلياً بالمرض في الكائنات الحية" هو ادعاء بيولوجي لا يثبته هذا المشروع. التحليل يظهر أن الكود يعمل وأن الخطوات صحيحة، لكنه لا يقول شيئاً عن الواقع البيولوجي. اختبار عملي: قبل أن تستنتج أي علاقة بيولوجية من نتائج المشروع، فحص تطبيقي: "هل هذه البيانات حقيقية أم اصطناعية؟ وهل هذا التحليل مصمم للإجابة عن سؤال بيولوجي حقيقي أم لتوضيح منهجية تحليلية؟"ـ

مثال تطبيقي على مشروع Brick by Brick:

· المدخلات: ملفات اصطناعية (تسلسلات FASTA، بيانات تعبير counts، بيانات وصفية)، سكربتات Bash و Python و R، ومعاملات التحليل (مثل عتبة الجودة، عتبة التعبير التفاضلي)ـ

· الافتراضات: أن البيانات الاصطناعية تمثل سيناريو تعليمي معقول، وأن السكربتات تعمل في البيئة المحددة، وأن المعاملات المختارة مناسبة للغرض التعليميـ

· المخرجات: تقارير تحليلية (إحصائيات التسلسلات، نتائج التعبير التفاضلي)، رسوم بيانية (Volcano، Heatmap، PCA، MA)، سجلات تشغيل، وتقرير نهائيـ

· القيود: المشروع تعليمي فقط، ولا يعتمد على بيانات حقيقية، ولا يمكن استخدام نتائجه للاستدلال البيولوجي. الفحوصات الأمنية مبسطة ولا تمثل نظاماً أمنياً تشغيلياً. التطبيع والاختبارات الإحصائية مبسطة مقارنة بالأنظمة الإنتاجية.

Applied example of the Brick by Brick project:

اختبار: قبل تشغيل المشروع، تحقق من وجود جميع الملفات المطلوبة، وتأكد من أن البيئة البرمجية متوافقة مع الإصدارات المحددة، وافهم أن النتائج تعكس سلوك الكود على بيانات اصطناعية، وليست استنتاجات بيولوجية.

· Inputs: synthetic files (FASTA sequences, expression counts, metadata), Bash/Python/R scripts, and analysis parameters (e.g., quality threshold, differential expression threshold).

· Assumptions: that the synthetic data represents a reasonable educational scenario, that the scripts work in the specified environment, and that the chosen parameters are appropriate for the educational purpose.

· Outputs: analytical reports (sequence statistics, differential expression results), plots (Volcano, Heatmap, PCA, MA), logs, and a final report.

· Limitations: the project is educational only, does not rely on real data, and its results cannot be used for biological inference. Security checks are simplified and do not represent an operational security system. Normalization and statistical tests are simplified compared to production systems.

Test: before running the project, verify that all required files exist, ensure the software environment is compatible with the specified versions, and understand that the results reflect code behavior on synthetic data, not biological conclusions.

عند تشغيل مشروع Brick by Brick، يجب توثيق ما يلي لضمان إعادة الإنتاج:

· الإصدار: إصدار Python، Biopython، R، والمكتبات المستخدمة (مثل pandas، numpy، ggplot2, DESeq2). يمكن تسجيل ذلك باستخدام pip freeze > requirements.txt في Python و sessionInfo() في R.

· المصدر: البيانات الاصطناعية (كيف أُنشئت، المعلمات المستخدمة، تاريخ الإنشاء)، والسكربتات المستخدمة (مصدرها، تاريخ التعديل).

· المعلمات: جميع المعاملات المستخدمة في التحليل (مثل عتبة الجودة في الفحوصات الأمنية، عتبة التعبير التفاضلي، طريقة التطبيع).

· التحويلات: خطوات المعالجة بالترتيب (توليد البيانات، تحليل التسلسلات، الفحوصات الأمنية، تحليل التعبير، توليد التقرير)، مع توثيق أي تعديلات على السكربتات أو البيانات.

When running the Brick by Brick project, the following must be documented to ensure reproducibility:

اختبار: فحص تطبيقي: "لو أعطيت هذا المشروع (مع جميع الملفات والسجلات) لزميل بعد عام، هل يستطيع إعادة تشغيله بالكامل والحصول على نفس النتائج دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Provenance: synthetic data (how it was created, parameters used, creation date), and scripts used (source, modification date).

· Version: Python, Biopython, R, and library versions (e.g., pandas, numpy, ggplot2, DESeq2). This can be recorded using pip freeze > requirements.txt in Python and sessionInfo() in R.

· Parameters: all parameters used in the analysis (e.g., quality threshold in security checks, differential expression threshold, normalization method).

· Transformations: processing steps in order (data generation, sequence analysis, security checks, expression analysis, report generation), with any modifications to scripts or data documented.

Test: Practical check: "If I gave this project (with all files and logs) to a colleague a year later, could they rerun it completely and obtain the same results without my help?" If the answer is no, the documentation is insufficient.

مشروع Brick by Brick بأكمله هو مثال على استخدام اختبارات اصطناعية قبل الاعتماد على بيانات حقيقية. جميع البيانات المستخدمة في المشروع هي اصطناعية، مما يسمح باختبار الكود وفهم سير العمل دون التعامل مع بيانات حساسة أو الاعتماد على اتصال بالإنترنت. هذه الممارسة تضمن أن الكود يعمل كما هو متوقع قبل تطبيقه على بيانات حقيقية. على سبيل المثال، قبل تحليل بيانات RNA-seq حقيقية، يمكن تشغيل المشروع على البيانات الاصطناعية للتحقق من أن سكربتات Bash تعمل، وأن تحليل Biopython يعطي النتائج المتوقعة، وأن الفحوصات الأمنية تصنف التسلسلات بشكل صحيح، وأن تحليل R ينتج الرسوم البيانية المطلوبة. إذا نجح المشروع على البيانات الاصطناعية، يمكن للباحث الانتقال إلى البيانات الحقيقية بثقة أكبر، مع العلم أن أي مشكلة مع البيانات الحقيقية تعكس خصائص البيانات وليس خطأ في الكود أو سير العمل.

The entire Brick by Brick project is an example of using synthetic tests before relying on real data. All data used in the project is synthetic, allowing testing of the code and understanding of the workflow without handling sensitive data or depending on internet connectivity. This practice ensures that the code works as

expected before applying it to real data. For example, before analyzing real RNA-seq data, the project can be run on synthetic data to verify that Bash scripts work, that Biopython analysis gives expected results, that security checks classify sequences correctly, and that R analysis produces the required plots. If the project succeeds on synthetic data, the researcher can move to real data with greater confidence, knowing that any problem with real data reflects the data's properties and not an error in the code or workflow.

اختبار عملي: قبل أن تطبق أي جزء من هذا المشروع على بيانات حقيقية، شغّل المشروع بالكامل على البيانات الاصطناعية، وتأكد من أن جميع المراحل تكتمل بنجاح، وأن المخرجات (التقارير، الرسوم البيانية، السجلات) تُنشأ بشكل صحيح. إذا نجح الاختبار، فأنت واثق من أن سير العمل يعمل كما هو متوقع، ويمكن تطبيقه على بيانات حقيقية مع إجراء التعديلات اللازمة (مثل تغيير مسارات الملفات، أو تعديل المعاملات لتناسب البيانات الحقيقية).

Practical test: before applying any part of this project to real data, run the entire project on synthetic data, and verify that all stages complete successfully, and that the outputs (reports, plots, logs) are generated correctly. If the test succeeds, the workflow is confident that the workflow works as expected, and the workflow can apply it to real data with the necessary adjustments (such as changing file paths, or adjusting parameters to suit the real data).

Chapter 56 — The Scientific Report

الفصل 56 — التقرير العلمي

Deeper Conceptual Explanation

الفكرة المركزية: السؤال، المنهج، النتائج، القيود، الاستشهاد، الإفصاح.

شرح أعمق للمفهوم

A scientific report is not merely a description of software. It is a structured argument: question, method, data, results, uncertainty, limitations, and an interpretation proportional to the evidence. Separating results from discussion helps prevent a computational statement from silently becoming a causal claim.

التقرير العلمي ليس وصفاً للبرنامج فقط. هو حجة منظمة: سؤال، منهج، بيانات، نتائج، عدم يقين، قيود، ثم تفسير يتناسب مع قوة الدليل. فصل النتائج عن المناقشة يساعد على منع انتقال عبارة حسابية إلى ادعاء سببي من دون سند.

Deeper Conceptual Explanation

شرح أعمق للمفهوم

1. The Scientific Report: Purpose and Audience

The scientific report is the document that conveys the project's research findings to the scientific community. Its audience includes other researchers, reviewers, funders, and sometimes policymakers. Therefore, the report must be clear, accurate, organized, and auditable. The primary purposes of the report are:

1. التقرير العلمي: الغرض والجمهور

التقرير العلمي هو الوثيقة التي تنقل نتائج بحثك إلى المجتمع العلمي. جمهوره هم الباحثون الآخرون، والمراجعون، والممولون، وأحياناً صانعو السياسات. لذلك، يجب أن يكون التقرير واضحاً، دقيقاً، ومنظماً، وقابلاً للتدقيق. الغرض الأساسي من التقرير هو:

· الإبلاغ: تقديم النتائج بشكل موضوعي.

· Reporting: presenting results objectively.

· الإقناع: إظهار أن الاستنتاجات مدعومة بالأدلة.

· Persuasion: showing that conclusions are supported by evidence.

· التوثيق: توفير سجل دائم للبحث يمكن للآخرين الرجوع إليه وبنائه عليه.

· Documentation: providing a permanent record that others can refer to and build upon.

· الشفافية: الكشف عن القيود، والافتراضات، ومصادر عدم اليقين.

· Transparency: disclosing limitations, assumptions, and sources of uncertainty.

مبدأ: التقرير العلمي الجيد يجيب عن أربعة أسئلة أساسية:

1. ما الذي فعلته؟ (المنهج)

2. ما الذي وجدته؟ (النتائج)

3. ماذا يعني ذلك؟ (التفسير)

4. ما الذي لا تعرفه؟ (القيود وعدم اليقين)

2. Structure of the Scientific Report

The scientific report follows a standard structure in most fields, consisting of main sections:

2. هيكل التقرير العلمي

التقرير العلمي يتبع هيكلاً قياسياً في معظم المجالات، يتكون من أقسام رئيسية:

العنوان (Title)

الملخص (Abstract)

يجب أن يكون العنوان موجزاً وواصفاً، ويعكس المحتوى الأساسي للتقرير. عنوان جيد يساعد القراء في فهم موضوع البحث بسرعة.

المقدمة (Introduction)

ملخص قصير (عادة 150-300 كلمة) يلخص السؤال، والمنهج، والنتائج الرئيسية، والاستنتاجات. الملخص هو أول ما يُقرأ، ووغالباً ما يحدد مدى وضوح التقرير ومضمونه العلمي.

تشرح المقدمة السياق العلمي، وتطرح السؤال البحثي، وتوضح لماذا هذا السؤال مهم. يجب أن تشير إلى المعرفة السابقة وتحدد الفجوة التي يسدها البحث.

المنهج (Methods)

يصف هذا القسم بالتفصيل كل ما فعلته: البيانات، والأدوات، والمعاملات، والخطوات التحليلية. يجب أن يكون مفصلاً بما يكفي ليتمكن باحث آخر من إعادة التجربة أو التحليل.

النتائج (Results)

المناقشة (Discussion)

يعرض هذا القسم النتائج بشكل موضوعي، دون تفسير. تستخدم الجداول والرسوم البيانية لتلخيص البيانات. النتائج يجب أن تكون منظمة ومنطقية.

هنا يتم تفسير النتائج في السياق العلمي. تُقارن النتائج مع الدراسات السابقة، وتُستخلص الاستنتاجات، ويُناقش معنى النتائج والقيود.

الاستنتاج (Conclusion)

المراجع (References)

ملخص قصير للاستنتاجات الرئيسية، مع إشارة إلى الآثار المترتبة والخطوات المستقبلية المحتملة.

Title

The title should be concise and descriptive, reflecting the main content of the report. A good title helps communicate the research topic quickly and clearly.

قائمة بجميع المصادر التي استشهدت بها في التقرير.

Abstract

A short summary (usually 150–300 words) summarizing the question, methods, main results, and conclusions. The abstract is the first thing that is seen, and often determines how clearly the report communicates its scientific content.

Introduction

The introduction explains the scientific context, presents the research question, and explains why this question is important. It should refer to prior knowledge and identify the gap that the research fills.

Methods

This section describes in detail everything you did: the data, tools, parameters, and analytical steps. It should be detailed enough for another researcher to repeat the experiment or analysis.

Results

This section presents the results objectively, without interpretation. Tables and figures are used to summarize data. Results should be organized and logical.

Discussion

Here, the results are interpreted in the scientific context. Results are compared with previous studies, conclusions are drawn, and the meaning of results and limitations are discussed.

Part 15 — Advanced Practice: Real Data, Statistical Models, Workflows, and AI-Assisted Biology

الباب 15 — الممارسة المتقدمة: البيانات الحقيقية، النماذج الإحصائية، إدارة سير العمل، والذكاء الاصطناعي في البيولوجيا

The previous chapters established a safe computational foundation. This part closes the most important gap between a controlled teaching environment and the conditions encountered in real bioinformatics: imperfect sequencing data, overdispersed counts, multi-step workflows, heterogeneous computing environments, operational security, and AI-assisted analysis.

Synthetic data remains valuable here, but it changes role: it becomes a test fixture used alongside real public datasets, not a substitute for them. The workflow should demonstrate both how to prove that a pipeline works on controlled inputs and how to diagnose why a real dataset violates the assumptions of that pipeline.

وضعت الفصول السابقة أساسًا حسابيًا آمنًا للتعلّم. أمّا هذا الباب فيغلق أهم فجوة بين البيئة التعليمية المضبوطة

والظروف التي تظهر في المعلوماتية الحيوية الواقعية: بيانات تسلسل غير مثالية، وعدّادات ذات تشتت زائد، وسير

عمل متعدد المراحل، وبيئات حوسبة مختلفة، وأمن تشغيلي، واستخدام أدوات الذكاء الاصطناعي مع بقاء التحقق العلمي مسؤولية الباحث.

Chapter 57 — From Clean Exercises to Real Sequencing Data

الفصل 57 — من التمارين النظيفة إلى بيانات التسلسل الحقيقية

تبقى البيانات الاصطناعية مفيدة هنا، لكن دورها يتغير: تصبح مجموعة اختبار مضبوطة تُستخدم إلى جانب مجموعات بيانات حقيقية عامة، لا بديلًا عنها. والمقصود هو تعلّم مسارين معًا: إثبات أن خط الأنابيب يعمل على مدخلات معروفة، وتشخيص الأسباب التي تجعل البيانات الحقيقية تخالف افتراضات ذلك الخط.

1. Why real data behave differently

A simulated FASTQ file can be generated with a known read length, a chosen error rate, and a known distribution. A real FASTQ file inherits the history of library preparation, sequencing chemistry, instrument performance, demultiplexing, filtering, and file conversion. Therefore, quality control is not a decorative first step; it is part of the scientific interpretation of the data.

1. لماذا تتصرف البيانات الحقيقية بصورة مختلفة؟

2. The real-data checklist

يمكن إنشاء ملف FASTQ محاكى بطول قراءات معروف، ومعدل أخطاء محدد، وتوزيع مضبوط. أما ملف FASTQ الحقيقي فيحمل تاريخ تحضير المكتبة، وكيمياء التسلسل، وأداء الجهاز، وفك الترميز، والترشيح، وتحويل الملفات. لذلك فإن ضبط الجودة ليس خطوة شكلية في بداية التحليل، بل جزء من تفسير البيانات علميًا.

Before analysis, record the sample sheet, sequencing platform, read layout, reference genome or transcriptome version, library preparation protocol when available, biological replicates, technical replicates, and any known batch variables. A count matrix without its provenance is not enough to reconstruct an experiment.

2. قائمة فحص البيانات الحقيقية

3. Quality control: adapters, quality scores, GC bias, and duplication

قبل التحليل يجب تسجيل جدول العينات، ومنصة التسلسل، ونمط القراءة، وإصدار الجينوم أو الترانسكريبتوم المرجعي، وبروتوكول تحضير المكتبة إن كان متاحًا، والتكرارات البيولوجية والتقنية، وأي متغيرات دفعة معروفة. فمصفوفة العدّ وحدها، من دون مصدرها وسياقها، لا تكفي لإعادة بناء التجربة.

Adapter contamination can create artificial similarity and misleading alignments. Per-base quality can deteriorate toward read ends. GC composition can differ from

the expected biological distribution because of library composition, amplification, sequencing chemistry, or selection. Duplicate reads may reflect genuine high abundance, optical duplication, or PCR amplification. None of these signals should be interpreted in isolation.

3. ضبط الجودة: الـAdapters ودرجات الجودة وانحياز GC والتكرارات

4. FastQC and MultiQC as evidence, not decoration

قد يؤدي تلوث القراءات بالـAdapters إلى تشابه مصطنع ومحاذاة مضللة. وقد تنخفض جودة القواعد تدريجيًا باتجاه نهاية القراءة. كما قد يختلف تركيب GC عن التوزيع المتوقع بيولوجيًا بسبب تركيب المكتبة أو التضخيم أو كيمياء التسلسل أو الانتقاء. وقد تمثل القراءات المتكررة وفرة حقيقية، أو تكرارًا بصريًا، أو تضخيمًا بواسطة PCR. ولا ينبغي تفسير أي إشارة من هذه الإشارات بمعزل عن بقية الأدلة.

Language / نوع الكود: **Bash / Shell**

```
# Run quality control on all FASTQ files in a directory.
# Keep the raw files unchanged and write reports to a separate directory.
mkdir -p qc/fastqc
fastqc --threads 4 --outdir qc/fastqc data/raw/*.fastq.gz
```

4. استخدام FastQC وMultiQC بوصفهما أدلة لا زخرفة

```
# Aggregate all FastQC reports into one project-level report.
mkdir -p qc/multiqc
multiqc qc/fastqc --outdir qc/multiqc
```

Language / نوع الكود: **Bash / Shell**

The commands preserve the raw data and separate quality-control outputs. In a real project, the exact FastQC and MultiQC versions should be recorded because the report is part of the provenance of the analysis.

5. Adapter trimming and the danger of automatic cleaning

يفصل هذا المثال البيانات الخام عن نواتج ضبط الجودة. وفي المشروع الحقيقي يجب تسجيل إصدارَي FastQC وMultiQC المستخدمين، لأن تقرير الجودة نفسه جزء من سلسلة مصدر النتائج وإمكانية إعادة إنتاج التحليل.

Trimming is not synonymous with improving data. An aggressive trimming policy can remove genuine biological sequence, shorten reads, alter mapping behavior, or preferentially remove one class of reads. A defensible workflow states why trimming is needed, which evidence triggered it, which parameters were used, and how many reads survived.

5. إزالة الـAdapters وخطر التنظيف الآلي

Language / نوع الكود: **Bash / Shell**

```
# Example with cutadapt.
# Replace the adapter sequence with the adapter reported by your library/QC documentation.
cutadapt \
-a ADAPTER_SEQUENCE \
-q 20 \
-m 30 \
-o data/trimmed/sample_R1.fastq.gz \
data/raw/sample_R1.fastq.gz
```

لا تعني إزالة الأجزاء الطرفية تلقائيًا أن البيانات أصبحت أفضل. فقد تؤدي سياسة إزالة شديدة إلى حذف تسلسل بيولوجي حقيقي، أو تقصير القراءات، أو تغيير سلوك المحاذاة، أو إزالة فئة معينة من القراءات بصورة انتقائية. لذلك يجب أن يوضح خط التحليل سبب الحاجة إلى الإزالة، والدليل الذي استدعاها، والمعاملات المستخدمة، وعدد القراءات التي بقيت بعد المعالجة.

Language / نوع الكود: **Bash / Shell**

The placeholder is deliberate: the correct adapter must come from the experimental protocol or quality-control evidence. A workflow should never copy a convenient adapter sequence merely because it appears in an example.

6. GC bias and PCR duplication require interpretation

وُضع اسم الـAdapter هنا كعنصر نائب عمدًا؛ فالـAdapter الصحيح يجب أن يأتي من بروتوكول التجربة أو من أدلة ضبط الجودة. ولا ينبغي نسخ تسلسل مناسب ظاهريًا من مثال تعليمي من دون التحقق من المكتبة الفعلية.

A GC curve is not automatically a defect. Biological samples can have non-uniform GC distributions, and different library types have different expectations. Likewise, duplicate reads are not automatically discarded in every sequencing analysis. Their meaning depends on assay design, library complexity, sequencing depth, and the downstream question.

6. انحياز GC وتكرار PCR يحتاجان إلى تفسير

7. Nanopore data introduce a different error profile

منحنى GC غير المعتاد لا يعني تلقائيًا وجود عيب. فقد تكون للعينات البيولوجية توزيعات GC غير متجانسة أصلًا، كما تختلف التوقعات باختلاف نوع المكتبة. وكذلك لا تُحذف القراءات المتكررة تلقائيًا في كل تحليل؛ إذ يعتمد معناها على تصميم التجربة وتعقيد المكتبة وعمق التسلسل والسؤال العلمي.

Long-read technologies change the balance between read length, error profile, structural variation, and downstream correction. A pipeline designed around short Illumina reads should not be copied unchanged. The first question is always what the platform and library preparation imply for the expected error structure.

7. بيانات Nanopore تقدم نمطًا مختلفًا من الأخطاء

8. From QC report to scientific decision

تغير تقنيات القراءات الطويلة العلاقة بين طول القراءة ونمط الأخطاء والتغيرات البنيوية والمعالجة اللاحقة. ولا ينبغي نسخ خط أنابيب صُمم لقراءات Illumina القصيرة كما هو. والسؤال الأول دائمًا هو: ماذا تعني منصة التسلسل وطريقة تحضير المكتبة بالنسبة إلى نمط الأخطاء المتوقع؟

The output of quality control is not a single score. It is a set of observations that must be connected to decisions. For example, severe adapter contamination may justify trimming; a consistent GC shift across all biological groups may be a property of the biology; a shift confined to one batch may instead indicate a technical problem. The decision belongs to the experimental context.

8. من تقرير الجودة إلى القرار العلمي

From real FASTQ to auditable measurements

ناتج ضبط الجودة ليس درجة واحدة. بل هو مجموعة ملاحظات يجب ربطها بقرارات. فقد يبرر تلوث شديد بالـAdapters إزالةً محددة؛ بينما قد يكون انحراف GC المتسق في جميع المجموعات خاصية بيولوجية؛ وإذا اقتصر الانحراف على دفعة واحدة فقد يشير إلى مشكلة تقنية. والقرار لا ينفصل عن سياق التجربة.

Real data are not ideal strings. Before downstream analysis, validate record structure, sequence/quality length equality, and non-empty input, then record summary statistics that can be compared with specialized QC reports. This example does not replace FastQC or MultiQC; it teaches how to build a small independent check that can itself be tested.

من FASTQ الحقيقي إلى قياسات قابلة للتدقيق

```
from pathlib import Path
import gzip
from dataclasses import dataclass
```

البيانات الحقيقية لا تُعامل كسلسلة نصية مثالية. قبل تشغيل أي تحليل لاحق، يجب التأكد من بنية كل سجل، وتساوي طول التسلسل والجودة، وعدم وجود ملف فارغ، ثم تسجيل إحصاءات يمكن مقارنتها بتقارير الأدوات المتخصصة. المثال التالي لا يحل محل FastQC أو MultiQC؛ وظيفته أن يعلّم بناء فحص مستقل بسيط يمكن اختباره.

```
@dataclass
class FastqStats:
```

records: int

bases: int

mean_length: float

mean_gc: float

mean_quality: float

```
def summarize_fastq(path: Path) -> FastqStats:
    records = 0
    bases = 0
    gc = 0
    quality_sum = 0
    quality_count = 0
with gzip.open(path, "rt", encoding="utf-8") as fh:
        while True:
            header = fh.readline()
            if not header:
```

break

```
seq = fh.readline().strip()
            plus = fh.readline().strip()
            qual = fh.readline().strip()
if not header.startswith("@") or not plus.startswith("+"):
                raise ValueError("Malformed FASTQ record")
            if len(seq) != len(qual):
                raise ValueError("Sequence/quality length mismatch")
```

records += 1

bases += len(seq)

gc += sum(base in "GCgc" for base in seq)

```
            scores = [ord(ch) - 33 for ch in qual]
```

quality_sum += sum(scores)

quality_count += len(scores)

```
if records == 0:
        raise ValueError("FASTQ contains no records")
return FastqStats(
        records=records,
```

```
        bases=bases,
        mean_length=bases / records,
        mean_gc=gc / bases if bases else 0.0,
        mean_quality=quality_sum / quality_count if quality_count else 0.0,
    )

stats = summarize_fastq(Path("data/raw/sample_R1.fastq.gz"))
print(stats)
```

The value is not the GC calculation itself but the independent validation layer. A large disagreement with a specialized QC tool is a reason to inspect the implementation, Phred interpretation, GC definition, or even whether the same file was analyzed. Independent checks reduce the chance that one mistake propagates through every stage.

القيمة هنا ليست في حساب GC وحده، بل في إنشاء طبقة تحقق مستقلة. إذا أعطت الأداة المتخصصة نتيجة مختلفة جذريًا عن هذا الفحص، فهذه إشارة إلى مراجعة الترميز، أو تفسير جودة Phred، أو طريقة حساب GC، أو الملف الذي عولج بالفعل. الاستقلال بين أدوات التحقق يقلل خطر تمرير الخطأ نفسه عبر جميع مراحل الخط.

Chapter 58 — RNA-seq Statistics Beyond the t-test

الفصل 58 — إحصاء RNA-seq بعد اختبار t

1. Why counts are not ordinary continuous measurements

1. لماذا لا تُعامل العدادات مثل القياسات المستمرة العادية؟

RNA-seq starts with counts. Counts are non-negative integers and their variance commonly changes with their mean. Replicates also vary biologically and technically. A simple t-test on transformed or simulated values can illustrate the idea of group comparison, but it is not an adequate replacement for a count model when analysing sequencing data.

يبدأ RNA-seq بعدّادات. والعدّادات أعداد صحيحة غير سالبة، كما أن التباين يتغير عادةً مع المتوسط، وتختلف التكرارات بيولوجيًا وتقنيًا. ويمكن لاختبار t على قيم محولة أو محاكاة أن يوضح فكرة مقارنة مجموعتين، لكنه ليس بديلًا مناسبًا عن نموذج للعدّادات عند تحليل بيانات التسلسل.

2. The negative binomial model

2. نموذج التوزيع السالب ذي الحدين

DESeq2 models count variability using the negative binomial distribution and estimates the dependence between mean and variance. edgeR likewise uses negative-binomial methodology and provides generalized linear models and quasi-

likelihood approaches. These methods are designed for replicated count experiments rather than treating each gene as a simple normal measurement.

يستخدم DESeq2 التوزيع السالب ذي الحدين لنمذجة تباين العدّادات ويقدّر العلاقة بين المتوسط والتباين. كما يستخدم edgeR منهجيات قائمة على التوزيع السالب ذي الحدين ويوفر نماذج خطية عامة وطرائق quasi-likelihood. وهذه الأساليب مصممة لتجارب العدّادات ذات التكرارات، بدل معاملة كل جين كما لو كان قياسًا طبيعيًا بسيطًا.

3. A conceptual GLM

3. النموذج الخطي العام من الناحية المفاهيمية

For a gene, the model connects expected counts to experimental variables through a link function. In simplified notation, the expected count is represented through a logarithmic link and includes a library-size or normalization component plus coefficients for biological conditions and covariates. The coefficient is not itself a p-value; it is an estimated effect on the model scale.

بالنسبة إلى جين معين، يربط النموذج العدّ المتوقع بمتغيرات التجربة عبر دالة ربط. وبصياغة مبسطة، يُمثَّل العدد المتوقع بواسطة رابط لوغاريتمي، مع مكوّن يتعلق بحجم المكتبة أو التطبيع، ومعاملات للظروف البيولوجية والمتغيرات المصاحبة. والمعامل نفسه ليس قيمة p؛ بل هو تقدير لتأثير على مقياس النموذج.

4. Dispersion is not an optional detail

Dispersion captures extra-Poisson variability. If two genes have similar mean counts but very different between-replicate variability, treating them as equally precise would distort inference. The dispersion estimate therefore contributes directly to how strongly the model distinguishes a systematic group effect from replicate-to-replicate noise.

4. التشتت ليس تفصيلًا ثانويًا

5. DESeq2 workflow with a real count matrix

يمثل التشتت مقدار التباين الإضافي فوق ما يتوقعه نموذج بواسون البسيط. فإذا كان لجينين متوسط عدّ متقارب، لكن تباين التكرارات بينهما مختلفًا بشدة، فإن معاملتهما على أنهما متساويا الدقة ستشوّه الاستدلال. لذلك يساهم تقدير التشتت مباشرةً في تمييز تأثير المجموعة المنتظم من الضوضاء بين التكرارات.

Language / نوع الكود: **R / Bioconductor**

5. سير عمل DESeq2 على مصفوفة عدّ حقيقية

```
library(DESeq2)
# counts: genes x samples integer count matrix
# coldata: one row per sample, with a condition column
coldata$condition <- factor(coldata$condition, levels = c("control", "treated"))
```

```
stopifnot(identical(colnames(counts), rownames(coldata)))
dds <- DESeqDataSetFromMatrix(
  countData = round(as.matrix(counts)),
  colData = coldata,
  design = ~ condition
)
dds <- DESeq(dds)
res <- results(dds, contrast = c("condition", "treated", "control"))
res <- res[order(res$padj), ]
head(as.data.frame(res))
```

وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. في R، أسماء الكائنات وبنية الـdata frame وصيغة النموذج أهم من مجرد نجاح السطر في التنفيذ. النتيجة الإحصائية لا تُفسر قبل التأكد من أن الصفوف والأعمدة والـmetadata مرتبطة بالعينات الصحيحة.

Language / نوع الكود: R / Bioconductor

This is a genuine DESeq2 workflow structure: sample metadata are matched to count-matrix columns, a design formula defines the comparison, DESeq2 fits its model, and results expose log2 fold changes, standard errors, test statistics, raw p-values, and adjusted p-values. The code requires an installed Bioconductor environment and cannot be claimed to have been executed in an environment where R is unavailable.

هذا هو الهيكل الفعلي لسير عمل DESeq2: تُطابق بيانات العينات مع أعمدة مصفوفة العدّ، وتحدد صيغة التصميم المقارنة، ثم يلائم DESeq2 النموذج، وتظهر في النتائج قيم log2 fold change والخطأ المعياري وإحصاء الاختبار وقيمة p والقيمة المصححة. ويحتاج الكود إلى بيئة Bioconductor مثبتة، لذلك لا يجوز الادعاء بأنه شُغّل في بيئة لا يتوافر فيها R.

6. Reading log2 fold change correctly

6. قراءة log2 fold change قراءة صحيحة

A log2 fold change of 1 corresponds to a two-fold ratio on the original scale; -1 corresponds to a half ratio. The estimate is directional and model-

dependent. It should not be read as a causal effect, and it should not be interpreted without considering uncertainty and the experimental design.

تعني قيمة log2 fold change تساوي 1 نسبة مقدارها ضعفين على المقياس الأصلي، بينما تعني القيمة -1 نصف النسبة. والتقدير اتجاهي ويتأثر بالنموذج والتصميم. ولا ينبغي قراءته بوصفه تأثيرًا سببيًا، ولا تفسيره بعيدًا عن عدم اليقين وتصميم التجربة.

7. Multiple testing is part of the model interpretation

7. الاختبارات المتعددة جزء من تفسير النموذج

Testing thousands of genes produces many opportunities for small p-values by chance. False discovery rate control addresses this multiplicity. An adjusted p-value does not mean that a gene has a particular probability of being biologically causal; it is a quantity attached to the statistical testing procedure.

إن اختبار آلاف الجينات يخلق فرصًا كثيرة لظهور قيم p صغيرة بالصدفة. ويعالج ضبط معدل الاكتشاف الكاذب هذه المشكلة. والقيمة p المصححة لا تعني أن الجين يمتلك احتمالًا معينًا لأن يكون سببيًا بيولوجيًا؛ بل هي كمية مرتبطة بإجراء الاختبار الإحصائي.

8. edgeR and quasi-likelihood

8. edgeR وQuasi-likelihood

edgeR provides negative-binomial models, generalized linear models, and quasi-likelihood methods. The quasi-likelihood framework gives an additional layer for modelling uncertainty in dispersion and is especially useful when the design contains multiple factors or when inference must account carefully for biological variability.

يوفر edgeR نماذج سلبية ذات حدين ونماذج خطية عامة وطرائق quasi-likelihood. ويضيف إطار quasi-likelihood طبقة إضافية لتمثيل عدم اليقين في التشتت، ويكون مفيدًا بصورة خاصة عندما يحتوي التصميم على عوامل متعددة أو عندما يتطلب الاستدلال تمثيل التباين البيولوجي بعناية.

9. Batch effects and covariates

9. تأثيرات الدفعات والمتغيرات المصاحبة

If sequencing was performed in two batches and treatment groups are perfectly separated by batch, the statistical model cannot magically recover the missing experimental balance. A design such as ~ batch + condition can adjust for a batch effect when the design contains information to estimate both effects. It cannot repair perfect confounding.

إذا أُجري التسلسل في دفعتين وكانت مجموعتا المعالجة منفصلتين تمامًا بحسب الدفعة، فلن يستطيع النموذج الإحصائي استعادة التوازن التجريبي المفقود بطريقة سحرية. ويمكن لتصميم مثل ~ batch + condition أن يضبط تأثير الدفعة عندما يحتوي التصميم على معلومات تسمح بتقدير التأثيرين، لكنه لا يستطيع إصلاح الالتباس الكامل بين العاملين.

10. What the t-test chapter should now mean

10. ما الذي ينبغي أن يعنيه فصل اختبار t الآن؟

The simple t-test example remains useful as a conceptual bridge: it teaches the idea of comparing groups and separating signal from noise. Its status must be explicit, however. It is a pedagogical model, not the recommended inferential engine for a standard RNA-seq count matrix. The advanced section now supplies the missing bridge to the models used in practice.

يبقى مثال اختبار t مفيدًا بوصفه جسرًا مفاهيميًا؛ فهو يوضح فكرة مقارنة المجموعات والفصل بين الإشارة والضوضاء. لكن يجب أن يكون وضعه واضحًا: هو نموذج تعليمي، وليس محرك الاستدلال الموصى به لمصفوفة عدّ RNA-seq القياسية. ويقدم هذا الباب المتقدم الجسر المفقود إلى النماذج المستخدمة في التطبيق العلمي.

A more complete DESeq2 workflow: from counts to effect estimates

تطبيق كامل نسبيًا لـ DESeq2: من مصفوفة العد إلى تقدير الأثر

This example makes the gap between a toy differential-expression demonstration and an actual count-based analysis explicit. Counts are linked to sample metadata, the design reflects the experiment, weakly supported genes are filtered, the model is fitted, an explicit contrast is extracted, and a documented shrinkage method is used when appropriate. Coefficient names should never be copied blindly; inspect the fitted model first.

هنا يظهر الفرق بوضوح بين «محاكاة فكرة التعبير التفاضلي» وبين تحليل counts فعلي. مصفوفة العدّ تُربط بجدول عينات، ويُكتب التصميم انطلاقًا من التجربة، ثم يُرشح الجينات ذات الدعم العددي الضعيف، ويُقدّر النموذج، ويُستخرج contrast محدد، ثم تُستخدم طريقة موثقة لتقليص تقدير log2 fold change عندما يكون ذلك مناسبًا. لا ينبغي نسخ أسماء المعاملات عميانًا؛ يجب فحصها من النموذج.

```
suppressPackageStartupMessages({
library(DESeq2)
library(ggplot2)
})
```

```
counts <- read.csv("results/gene_counts.csv", row.names = 1, check.names = FALSE)
coldata <- read.csv("metadata/samples.csv", row.names = 1)
coldata$batch <- factor(coldata$batch)
dds <- DESeqDataSetFromMatrix(
countData = round(as.matrix(counts)),
colData = coldata,
design = ~ batch + condition
)
keep <- rowSums(counts(dds) >= 10) >= 2
dds <- dds[keep, ]
dds <- DESeq(dds)
res <- results(
dds,
contrast = c("condition", "treated", "control"),
alpha = 0.05
)
res <- lfcShrink(
dds,
coef = "condition_treated_vs_control",
type = "apeglm"
)
write.csv(
as.data.frame(res),
"results/deseq2_treated_vs_control.csv"
)
```

The t-test has not disappeared from statistics; it has simply been placed in the right context. It can teach hypotheses and mean differences in suitable settings, but it is not the model that directly represents RNA-seq count behavior. Here dispersion, library size, design, residual variation, and multiple testing are handled within one framework.

لاحظ أن اختبار t لم يختفِ من الإحصاء الحيوي؛ بل أصبح في مكانه الصحيح. يمكن استخدامه لتعليم فكرة الفرضية والفرق المتوسط في سياقات مناسبة، لكنه ليس النموذج الذي يصف مباشرةً طبيعة counts في

RNA-seq. هنا تظهر dispersion، وحجم المكتبة، والتصميم، والتباين المتبقي، وتعدد الاختبارات ضمن إطار واحد.

Chapter 59 — Real Reproducible Workflows: Snakemake, Nextflow, Conda, Docker, and Apptainer

الفصل 59 — سير العمل القابل لإعادة الإنتاج: Snakemake وNextflow وConda وDocker وApptainer

1. Why a collection of Bash commands is not yet a workflow system

1. لماذا لا تكفي مجموعة أوامر Bash لتصبح نظام سير عمل؟

Bash remains excellent for small, transparent tasks. The limitation appears when dependencies, parallelism, caching, provenance, retries, portability, and dozens of samples become part of the problem. A workflow manager expresses relationships between inputs and outputs so that the system can determine what needs to run and what can be reused.

يبقى Bash ممتازًا للمهام الصغيرة والواضحة. وتظهر حدوده عندما تدخل الاعتماديات والتوازي والتخزين المؤقت ومصدر النتائج وإعادة المحاولة وقابلية النقل وعشرات العينات في المشكلة. ويعبّر مدير سير العمل عن العلاقات بين المدخلات والمخرجات بحيث يستطيع النظام تحديد ما يجب تشغيله وما يمكن إعادة استخدامه.

2. Snakemake: rules and dependencies

```
rule trim_reads:
    input:
        "data/raw/{sample}.fastq.gz"
    output:
        "data/trimmed/{sample}.fastq.gz"
    shell:
        "cutadapt -q 20 -m 30 -o {output} {input}"
```

2. Snakemake: القواعد والاعتماديات

```
rule qc:
    input:
        "data/trimmed/{sample}.fastq.gz"
    output:
        "qc/{sample}.html"
    shell:
```

"fastqc --outdir qc {input}"

A production workflow requires more than a successful step definition. Versions, resources, inputs, outputs, and container identities should be explicit, with image digests recorded when containers provide part of the reproducibility guarantee. Failures should remain diagnosable and restartable without unnecessary recomputation.

سير العمل الحقيقي يحتاج أكثر من تعريف خطوة ناجحة. يجب تثبيت الإصدارات، وتحديد الموارد، وتسجيل المدخلات والمخرجات، وتثبيت صورة الحاوية أو digest عندما تكون الحاويات جزءًا من الضمان. كما يجب أن تكون حالة الفشل قابلة للتشخيص وإعادة التشغيل دون إعادة حساب غير ضروري.

The important idea is not the shell command itself. Snakemake records the dependency graph: qc depends on trimmed reads. With many samples, the same rule can be instantiated for each sample, and the workflow engine can schedule independent jobs in parallel.

3. Nextflow: channels and processes

الفكرة الأساسية هنا ليست أمر الـShell نفسه، بل رسم الاعتماديات: فحص الجودة يعتمد على القراءات بعد الإزالة. ومع تعدد العينات يمكن تطبيق القاعدة نفسها على كل عينة، ثم يستطيع مدير سير العمل جدولة المهام المستقلة بالتوازي.

Language / نوع الكود: **Bash / Shell**

```
nextflow.enable.dsl=2

3. Nextflow: القنوات والعمليات

process FASTQC {
  input:
    path reads
  output:
    path "*.html"
  script:
    """
    fastqc --outdir . ${reads}
    """
}

workflow {
  reads = channel.fromPath('data/trimmed/*.fastq.gz')
  FASTQC(reads)
```

```
}
```

Language / نوع الكود: **Nextflow DSL2**

Nextflow models data movement between processes. The same workflow can be adapted to local execution, HPC schedulers, and containerized environments. The process boundary also makes resource requirements and execution provenance easier to manage.

4. Conda is environment management, not the whole reproducibility story

يمثل Nextflow انتقال البيانات بين العمليات. ويمكن تكييف خط العمل نفسه للتنفيذ المحلي أو على أنظمة الحوسبة عالية الأداء أو داخل حاويات. كما تساعد حدود العمليات على إدارة الموارد وسجل التنفيذ بصورة أوضح.

Conda can isolate package versions and provide a practical environment specification. It does not by itself record every operating-system detail, external binary, reference database version, or workflow decision. Reproducibility is a stack: code, dependencies, containers where appropriate, reference data, parameters, provenance, and execution environment.

4. Conda لإدارة البيئة وليس قصة إعادة الإنتاج كلها

5. Docker and OCI containers

يمكن لـConda عزل إصدارات الحزم وتوفير وصف عملي للبيئة. لكنه لا يسجل بمفرده كل تفاصيل نظام التشغيل أو البرامج الثنائية الخارجية أو إصدار قواعد البيانات المرجعية أو قرارات سير العمل. وإمكانية إعادة الإنتاج طبقات: الكود، والاعتماديات، والحاويات عند الحاجة، والبيانات المرجعية، والمعاملات، ومصدر النتائج، وبيئة التنفيذ.

5. Docker وحاويات OCI

Language / نوع الكود: **Code** / **Command**

```
FROM ubuntu:24.04
RUN apt-get update \
&& apt-get install -y --no-install-recommends fastqc \
&& rm -rf /var/lib/apt/lists/*
WORKDIR /work
ENTRYPOINT ["fastqc"]
```

Language / نوع الكود: **Dockerfile**

A container image can package software and its runtime dependencies. For scientific reproducibility, pinning an image by digest is stronger than relying

only on a mutable tag. The image is still not the entire experiment: reference files and input data must also be versioned and identified.

يمكن لصورة الحاوية أن تجمع البرنامج واعتماديات وقت التشغيل. ولإعادة الإنتاج العلمي يكون تثبيت الصورة بواسطة digest أقوى من الاعتماد على اسم tag قابل للتغير. ومع ذلك فالحاوية ليست التجربة كلها؛ إذ يجب أيضًا إصدار الملفات المرجعية وبيانات الإدخال وتعريفها.

6. Apptainer on HPC systems

6. Apptainer على أنظمة الحوسبة عالية الأداء

Apptainer is designed for portable containers in environments such as shared HPC systems. Its SIF images can be signed and verified, and its documentation explicitly addresses reproducible builds. This makes it useful when a workflow needs container isolation without assuming that Docker is the execution model of a shared cluster.

صُمم Apptainer للحاويات المحمولة في بيئات مثل أنظمة الحوسبة عالية الأداء المشتركة. ويمكن توقيع صور SIF والتحقق منها، كما توثق أداته خيارات للبناء القابل لإعادة الإنتاج. لذلك يكون مفيدًا عندما يحتاج خط العمل إلى عزل البيئة دون افتراض أن Docker هو نموذج التنفيذ المناسب للعنقود المشترك.

7. The modern workflow stack

A practical modern stack can be described as: workflow manager for dependencies and scheduling; environment manager or container for software isolation; version control for code; immutable or versioned references for data; structured configuration for parameters; continuous integration for tests; and a provenance record connecting all of these pieces.

7. طبقات سير العمل الحديثة

8. A minimal reproducible project layout

يمكن وصف الطبقات الحديثة لسير العمل على النحو الآتي: مدير سير العمل للاعتماديات والجدولة؛ مدير بيئة أو حاوية لعزل البرامج؛ نظام تحكم في الإصدارات للكود؛ مراجع ثابتة أو ذات إصدارات للبيانات؛ ملف إعداد منظم للمعاملات؛ تكامل مستمر للاختبارات؛ وسجل مصدر نتائج يربط كل هذه الأجزاء.

Language / نوع الكود: **Code** / **Command**

```
project/
├── README.md
├── config/
```

```
│   └── config.yaml
├── data/
│   ├── raw/          # never modified in place
│   └── processed/
├── workflow/
│   ├── Snakefile
│   ├── rules/
│   └── scripts/
├── envs/
│   └── rnaseq.yaml
├── containers/
│   └── Dockerfile
├── results/
├── logs/
└── tests/
```

8. بنية دنيا لمشروع قابل لإعادة الإنتاج

Language / نوع الكود: **Text** / **Project Structure**

The structure separates immutable raw data, workflow logic, environments, results, logs, and tests. The exact directories can vary, but the principle is to make every dependency visible rather than hiding it in a sequence of undocumented shell commands.

تفصل هذه البنية البيانات الخام غير القابلة للتعديل عن منطق سير العمل والبيئات والنتائج والسجلات والاختبارات. ويمكن تغيير أسماء المجلدات، لكن المبدأ هو جعل الاعتماديات مرئية بدل إخفائها داخل سلسلة أوامر Shell غير موثقة.

9. CI and workflow testing

A workflow should have a small test dataset and expected properties. Tests should verify that files are produced, schemas are correct, sample identifiers remain aligned, and obvious failure states stop the workflow. A green test does not prove biological validity; it proves that specified computational invariants hold.

9. التكامل المستمر واختبار سير العمل

Chapter 60 — From Educational Biosecurity to Operational Defensive Engineering

الفصل 60 — من نموذج الأمن الحيوي التعليمي إلى الهندسة الدفاعية التشغيلية

ينبغي أن يحتوي سير العمل على مجموعة اختبار صغيرة وخصائص متوقعة. ويجب أن تتحقق الاختبارات من إنتاج الملفات، وصحة المخططات، وبقاء معرفات العينات متطابقة، وتوقف الخط عند حالات الفشل الواضحة. ونجاح الاختبار لا يثبت صحة بيولوجية؛ بل يثبت تحقق خصائص حسابية محددة.

1. The boundary of biobrick_security

1. حدود biobrick_security

The educational package is valuable precisely because it isolates concepts such as fail-closed behavior, audit logging, human review, policy separation, and unit testing. It is not a production biosecurity platform. A real system must add identity, authorization, secure deployment, secrets management, monitoring, incident response, policy governance, data protection, and independent review.

2. A defensive architecture

تكمن قيمة الحزمة التعليمية في أنها تعزل مفاهيم مثل الفشل المغلق، وسجلات التدقيق، والمراجعة البشرية، وفصل السياسة عن المنطق، واختبارات الوحدة. لكنها ليست منصة أمن حيوي تشغيلية. فالنظام الحقيقي يحتاج إلى الهوية والصلاحيات والنشر الآمن وإدارة الأسرار والمراقبة والاستجابة للحوادث وحوكمة السياسات وحماية البيانات والمراجعة المستقلة.

A production-oriented defensive service should separate request intake, policy evaluation, computational analysis, human review, decision recording, and audit export. No single classifier should be the final authority. The system should also distinguish a technical failure from a policy rejection and from an unresolved scientific ambiguity.

2. بنية دفاعية متعددة الطبقات

3. Fail-closed is necessary but not sufficient

ينبغي للنظام الدفاعي الموجه للإنتاج أن يفصل بين استقبال الطلب، وتقييم السياسة، والتحليل الحاسوبي، والمراجعة البشرية، وتسجيل القرار، وتصدير سجل التدقيق. ولا ينبغي أن يكون مصنف واحد هو صاحب القرار النهائي. كما يجب التمييز بين الفشل التقني والرفض السياسي أو الإجرائي والحالة العلمية غير المحسومة.

Fail-closed means that a missing prerequisite does not silently become permission. It does not solve compromised credentials, malicious insiders, incorrect policies,

stale references, model drift, or poor human review. Operational security therefore requires independent controls that can fail differently from one another.

3. الفشل المغلق ضروري لكنه غير كافٍ

4. Policy as versioned data

يعني الفشل المغلق أن غياب شرط مسبق لا يتحول تلقائيًا إلى إذن. لكنه لا يحل مشكلة بيانات اعتماد مخترقة أو مستخدم داخلي خبيث أو سياسة غير صحيحة أو مراجع قديمة أو انجراف النموذج أو مراجعة بشرية ضعيفة. لذلك يحتاج الأمن التشغيلي إلى ضوابط مستقلة لا تفشل بالطريقة نفسها.

Language / نوع الكود: **Code** / **Command**

```
policy_version: "2026-01"
mode: "review_required"
fail_closed: true
required_controls:
- authenticated_user
- approved_project
- current_reference_version
- audit_logging
review:
required_for: [ambiguous, high_risk]
record_decision: true
```

4. السياسة بوصفها بيانات ذات إصدار

Language / نوع الكود: **YAML**

A policy file makes governance explicit and reviewable. The values shown here are intentionally abstract; the goal is to demonstrate configuration and provenance, not to encode operational biological screening rules.

5. Software supply-chain security

يجعل ملف السياسة الحوكمة صريحة وقابلة للمراجعة. والقيم المعروضة هنا مجردة عمدًا؛ فالهدف هو تعليم الإعداد والتتبع ومصدر القرار، لا ترميز قواعد تشغيلية للفحص البيولوجي.

A production system should know which packages and container images it runs. Software bills of materials, pinned dependencies, signed artifacts, vulnerability scanning, code review, least privilege, and controlled

deployment reduce the risk that a seemingly harmless update changes the behavior of a security-sensitive component.

5. أمن سلسلة توريد البرمجيات

6. Audit logs should answer five questions

يجب أن يعرف النظام التشغيلي الحزم وصور الحاويات التي يعمل بها. وتساعد قوائم مكونات البرمجيات، وتثبيت الاعتماديات، وتوقيع القطع البرمجية، وفحص الثغرات، ومراجعة الكود، وأقل صلاحية ممكنة، والنشر المقيد في تقليل خطر أن يغير تحديث يبدو عاديًا سلوك مكوّن حساس أمنيًا.

A useful audit record should make it possible to reconstruct who initiated an action, what input or request was considered, which policy and model versions were used, what decision was made, and who approved or reviewed it. The record should be append-only or otherwise tamper-evident and should not contain unnecessary sensitive sequence content.

6. يجب أن تجيب سجلات التدقيق عن خمسة أسئلة

7. Human review is an engineering component

يجب أن يسمح سجل التدقيق المفيد بإعادة بناء من بدأ الإجراء، وما المدخل أو الطلب الذي نوقش، وأي إصدار من السياسة والنموذج استُخدم، وما القرار الذي اتُخذ، ومن وافق عليه أو راجعه. وينبغي أن يكون السجل غير قابل للتعديل إلا بطريقة تكشف التغيير، وألا يحتوي على تسلسلات حساسة غير ضرورية.

Human-in-the-loop design fails when the human is treated as an unmeasured button at the end of the pipeline. A real review process needs criteria, training, escalation paths, time limits, conflict-of-interest rules, recorded rationales, and periodic quality assessment. The goal is not to make humans rubber-stamp algorithmic decisions, but to give them enough context to challenge the system.

7. المراجعة البشرية مكوّن هندسي

8. Roadmap from educational package to production service

يفشل تصميم الإنسان داخل الحلقة عندما يُعامل المراجع البشري كزر غير مقاس في نهاية خط العمل. تحتاج المراجعة الحقيقية إلى معايير وتدريب ومسارات تصعيد وحدود زمنية وقواعد تضارب المصالح وتسجيل أسباب القرار وتقييم دوري للجودة. والهدف ليس أن يصدق الإنسان قرار الخوارزمية تلقائيًا، بل أن يمتلك السياق الكافي لمراجعتها والاعتراض عليها.

Language / نوع الكود: **Code** / **Command**

Phase 1: threat model and governance

Phase 2: identity, authorization, and audit trail

Phase 3: policy versioning and review workflow

Phase 4: isolated execution and dependency pinning

Phase 5: automated tests and adversarial robustness tests

Phase 6: independent security review

Phase 7: staged deployment with monitoring and rollback

Phase 8: periodic policy, model, and dependency review

8. خريطة الطريق من الحزمة التعليمية إلى الخدمة التشغيلية

Language / نوع الكود: **Text** / **Roadmap**

This roadmap is intentionally defensive and governance-oriented. It does not provide biological instructions. It describes the engineering work required to move from a teaching prototype to a controlled service.

هذه الخريطة دفاعية وموجهة إلى الحوكمة عمدًا، ولا تقدم تعليمات بيولوجية تشغيلية. وهي تصف العمل الهندسي المطلوب للانتقال من نموذج تعليمي إلى خدمة مضبوطة.

Chapter 61 — Generative AI in Daily Bioinformatics Practice

الفصل 61 — الذكاء الاصطناعي التوليدي في الممارسة اليومية للمعلوماتية الحيوية

1. AI is a tool, not a source of scientific authority

Modern AI systems can generate code, explain errors, summarize documentation, draft tests, transform data formats, and help explore hypotheses. They can also hallucinate APIs, invent references, silently change assumptions, or produce code that runs while being scientifically wrong. The correct mental model is an assistant whose outputs require verification.

1. الذكاء الاصطناعي أداة وليس سلطة علمية

2. Using coding assistants safely

يمكن لأنظمة الذكاء الاصطناعي الحديثة توليد الكود وشرح الأخطاء وتلخيص الوثائق وكتابة الاختبارات وتحويل صيغ البيانات والمساعدة في استكشاف الفرضيات. لكنها قد تهلوس واجهات برمجية، أو تخترع مراجع، أو تغير افتراضات من دون تنبيه، أو تنتج كودًا يعمل تقنيًا وهو خاطئ علميًا. والنموذج الصحيح هو أنها مساعد تحتاج مخرجاته إلى تحقق.

```
def gc_fraction(seq: str) -> float:
    """Return the fraction of G/C bases in a DNA sequence."""
    seq = seq.upper().replace(" ", "")
```

```
  if not seq:
      raise ValueError("Sequence must not be empty")
  valid = set("ACGTN")
  if any(base not in valid for base in seq):
      raise ValueError("Unexpected base in sequence")
  informative = sum(base in "ACGT" for base in seq)
  if informative == 0:
      return float("nan")
  return sum(base in "GC" for base in seq) / informative
```

2. استخدام مساعدات البرمجة بأمان

```
assert abs(gc_fraction("ATGC") - 0.5) < 1e-12
```

الدوال المعرفة هنا (gc_fraction) تمثل حدودًا واضحة بين المدخلات والتحويل والنتيجة. القيمة العلمية لهذا الفصل ليست في اختصار الكود، بل في جعل الافتراضات التي تدخل إلى الدالة قابلة للاختبار وإعادة الاستخدام. وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة.

An AI assistant can propose a function like this, but the scientific contract must come first: what does GC fraction mean when N bases occur, what should an empty sequence do, and what inputs are valid? Tests turn the contract into something the computer can check.

3. A verification protocol for AI-generated code

يمكن لمساعد ذكاء اصطناعي أن يقترح دالة كهذه، لكن العقد العلمي يجب أن يسبق الاقتراح البرمجي: ماذا تعني نسبة GC عند وجود N؟ ماذا يحدث للتسلسل الفارغ؟ وما المدخلات المقبولة؟ وتحول الاختبارات هذا العقد إلى شروط يستطيع الحاسوب التحقق منها.

Every generated code block should pass at least four checks: syntax and execution, expected-output tests, boundary cases, and scientific interpretation. For data analysis, a fifth check is required: compare the method with the authoritative documentation or primary methodological paper. Passing a unit test cannot prove that the method answers the biological question correctly.

3. بروتوكول للتحقق من الكود المولّد بالذكاء الاصطناعي

4. AlphaFold as a practical example of AI-enabled biology

ينبغي أن يمر كل كود مولّد بالذكاء الاصطناعي بأربعة اختبارات على الأقل: صحة الصياغة والتنفيذ، واختبارات الناتج المتوقع، وحالات الحدود، والتفسير العلمي. وفي تحليل البيانات يلزم اختبار خامس: مقارنة الطريقة بالوثائق الرسمية أو الورقة المنهجية الأصلية. فنجاح اختبار الوحدة لا يثبت أن الطريقة تجيب عن السؤال البيولوجي بصورة صحيحة.

AlphaFold demonstrated that machine learning can produce highly accurate protein structure predictions for many proteins, and the AlphaFold Protein Structure Database made a large body of predictions openly accessible. AlphaFold 3 and the AlphaFold Server extend this ecosystem toward interactions involving proteins and other biomolecules. The scientific use remains hypothesis generation and structural interpretation, not a replacement for experimental evidence.

4. AlphaFold مثال عملي على البيولوجيا المدعومة بالذكاء الاصطناعي

5. A safe structural-bioinformatics workflow

أظهر AlphaFold أن التعلم الآلي يستطيع إنتاج تنبؤات عالية الدقة لبنية كثير من البروتينات، وأتاح AlphaFold Protein Structure Database عددًا ضخمًا من التنبؤات بصورة مفتوحة. كما وسّع AlphaFold 3 وAlphaFold Server هذا المجال نحو التفاعلات التي تشمل البروتينات وجزيئات حيوية أخرى. ومع ذلك يبقى الاستخدام العلمي في نطاق توليد الفرضيات وتفسير البنية، وليس بديلًا عن الدليل التجريبي.

Language / نوع الكود: **Code** / **Command**

Sequence

↓

Quality and identity checks

↓

Structure prediction / retrieval

↓

Confidence inspection

↓

Domain and interface interpretation

↓

Hypothesis generation

↓

Independent evidence

↓

Experimental or literature validation

5. سير عمل آمن للمعلوماتية الحيوية البنيوية

Language / نوع الكود: **Text** / **Workflow**

The critical step is the transition from prediction to independent evidence. A confident structural prediction can make a hypothesis more plausible; it does not establish biological function, mechanism, or clinical efficacy by itself.

6. AI-assisted literature review

الخطوة الحاسمة هي الانتقال من التنبؤ إلى الدليل المستقل. فقد يجعل التنبؤ البنيوي عالي الثقة فرضية ما أكثر معقولية، لكنه لا يثبت وحده الوظيفة البيولوجية أو الآلية أو الفاعلية السريرية.

An AI system can help turn a broad question into search terms, cluster papers by topic, or draft a comparison table. It should not be treated as the bibliographic record itself. The final citation must be opened, checked, and linked to the exact claim it supports. Persistent identifiers such as DOI and PMID remain the authoritative anchors.

6. استخدام الذكاء الاصطناعي في مراجعة الأدبيات

يمكن لنظام ذكاء اصطناعي أن يساعد في تحويل سؤال واسع إلى عبارات بحث، أو تجميع الأوراق حسب الموضوع، أو إعداد جدول مقارنة أولي. لكنه لا ينبغي أن يكون هو السجل الببليوغرافي نفسه. يجب فتح المرجع النهائي والتحقق منه وربطه بالادعاء المحدد الذي يدعمه. وتبقى المعرفات الدائمة مثل DOI وPMID مراسي أساسية للمصدر.

7. AI and privacy

Before sending biological data to an external AI service, determine whether the data are public, de-identified, proprietary, clinically sensitive, or otherwise restricted. A convenient prompt is not a justification for transferring sensitive sequences or metadata. Local models, approved enterprise services, redaction, or synthetic test data may be safer alternatives depending on the governance context.

7. الذكاء الاصطناعي والخصوصية

8. AI-generated code in regulated or high-consequence contexts

قبل إرسال بيانات بيولوجية إلى خدمة ذكاء اصطناعي خارجية يجب تحديد ما إذا كانت البيانات عامة أو منزوعة الهوية أو مملوكة أو حساسة سريريًا أو خاضعة لقيود أخرى. ولا يكفي أن يكون إرسال البيانات سهلًا لتبرير نقل تسلسلات أو بيانات وصفية حساسة. وقد تكون النماذج المحلية أو الخدمات المؤسسية المعتمدة أو إزالة التفاصيل الحساسة أو البيانات الاصطناعية بدائل أكثر أمانًا بحسب سياق الحوكمة.

When software influences clinical, security, or other high-consequence decisions, generated code needs stronger controls: review by qualified developers, traceable changes, automated tests, dependency scanning, reproducible builds, and documented approval. The model that generated a line of code is not a substitute for accountability.

8. الكود المولّد بالذكاء الاصطناعي في السياقات المنظمة أو عالية العواقب

Chapter 62 — The Real-Data Capstone: From FASTQ to Defensible Conclusion

الفصل 62 — المشروع التطبيقي بالبيانات الحقيقية: من FASTQ إلى استنتاج يمكن الدفاع عنه

عندما يؤثر البرنامج في قرارات سريرية أو أمنية أو قرارات أخرى عالية العواقب، يحتاج الكود المولّد إلى ضوابط أقوى: مراجعة من مطورين مؤهلين، وتغييرات قابلة للتتبع، واختبارات آلية، وفحص للاعتماديات، وبناء قابل لإعادة الإنتاج، وموافقة موثقة. والنموذج الذي ولّد سطرًا من الكود ليس بديلًا عن المسؤولية.

1. The capstone now has two lanes

The first lane remains the controlled educational lane: synthetic data with known properties and expected outputs. The second lane is the realism lane: a small public dataset with recorded provenance, real quality variation, and documented deviations from the assumptions of the teaching examples. The same code should not be assumed to work unchanged; the pipeline must earn that claim through tests and QC.

1. أصبح للمشروع مساران متكاملان

2. Project manifest

يبقى المسار الأول هو المسار التعليمي المضبوط: بيانات اصطناعية ذات خصائص معروفة ومخرجات متوقعة. أما المسار الثاني فهو مسار الواقعية: مجموعة بيانات عامة صغيرة ذات مصدر موثق وتباين جودة حقيقي وانحرافات

موثقة عن افتراضات الأمثلة التعليمية. ولا ينبغي افتراض أن الكود نفسه سيعمل دون تعديل؛ بل يجب أن يثبت خط العمل ذلك عبر الاختبارات وضبط الجودة.

Language / نوع الكود: **Code** / **Command**

2. سجل المشروع

project: brick_by_brick_real_data_capstone

analysis_type: rnaseq

input_source: public_repository

reference:

genome: "record exact assembly/version"

annotation: "record exact annotation release"

software:

workflow_manager: "record version"

aligner_or_quantifier: "record version"

statistics: "DESeq2 or edgeR, record version"

qc:

raw_reads: true

post_trim_reads: true

provenance:

accession: "record accession"

downloaded_at: "record timestamp"

checksum: "record checksum"

Language / نوع الكود: **YAML**

The manifest forces the project to state what it knows and what it still needs to verify. The placeholder values are preferable to invented accession numbers or versions because reproducibility begins with refusing to fabricate provenance.

3. The raw-data branch

يجبر هذا السجل المشروع على توضيح ما يعرفه وما يحتاج إلى التحقق منه. وتُعد القيم النائبة أفضل من اختراع أرقام وصول أو إصدارات؛ لأن إعادة الإنتاج تبدأ برفض اختلاق مصدر للبيانات.

Raw data should be treated as immutable evidence. Download it once, record its checksum, and never modify it in place. All cleaning and trimming should create new files. This distinction is scientifically important because it preserves the ability to compare the original observation with every processed derivative.

3. فرع البيانات الخام

ينبغي التعامل مع البيانات الخام بوصفها دليلًا غير قابل للتعديل. تُنزّل مرة واحدة، ويسجل checksum الخاص بها، ولا تُعدل في مكانها. ويجب أن تنشئ عمليات التنظيف والإزالة ملفات جديدة. وهذا الفصل مهم علميًا لأنه يحافظ على القدرة على مقارنة الملاحظة الأصلية بكل مشتق تمت معالجته.

Language / نوع الكود: **Bash / Shell**

```
# Record a checksum for a downloaded FASTQ file.
sha256sum data/raw/sample_R1.fastq.gz > provenance/sample_R1.sha256

# Record file size and timestamp without modifying the file.
stat data/raw/sample_R1.fastq.gz > provenance/sample_R1.stat
```

Language / نوع الكود: **Bash / Shell**

Checksums provide an identity for a file at a specific point in time. They do not prove that the biological sample was collected correctly, but they help prove that the computational input file has not silently changed.

يوفر checksum هوية للملف في لحظة محددة. وهو لا يثبت أن العينة البيولوجية جُمعت بطريقة صحيحة، لكنه يساعد على إثبات أن ملف الإدخال الحسابي لم يتغير بصمت.

4. The QC gate

The workflow should not proceed simply because a file exists. A QC gate can require that reports are present, sample identifiers match metadata, read counts are within documented expectations, and serious warnings have been reviewed. Some warnings may be accepted with justification; others should stop the workflow.

4. بوابة ضبط الجودة

5. Quantification and the statistical branch

لا ينبغي أن ينتقل سير العمل لمجرد وجود الملف. يمكن لبوابة ضبط الجودة أن تشترط وجود التقارير، وتطابق معرفات العينات مع البيانات الوصفية، ووقوع أعداد القراءات ضمن التوقعات الموثقة، ومراجعة التحذيرات المهمة. وقد تُقبل بعض التحذيرات مع توثيق سبب قبولها، بينما ينبغي أن توقف تحذيرات أخرى سير العمل.

Once reads have passed QC, the project must state whether the biological question requires alignment-based quantification or transcript-level pseudoalignment or another validated strategy. The resulting count matrix is then connected to sample metadata and the statistical model. This is where the earlier chapters on data structures, provenance, and experimental design become necessary rather than decorative.

5. التقدير وفرع التحليل الإحصائي

6. Biological interpretation is a separate layer

بعد اجتياز القراءات ضبط الجودة يجب تحديد ما إذا كان السؤال البيولوجي يتطلب تقديرًا قائمًا على المحاذاة أو تقديرًا على مستوى الترانسكريبت أو استراتيجية أخرى موثقة. ثم تُربط مصفوفة العد ببيانات العينات وبالنموذج الإحصائي. وهنا تصبح الفصول السابقة عن هياكل البيانات ومصدر النتائج وتصميم التجربة ضرورية، لا مجرد إضافات شكلية.

A statistically significant gene is not automatically an important biological target. Interpretation should integrate effect size, uncertainty, annotation quality, pathway context, prior evidence, experimental design, and independent validation. The final report should distinguish what the analysis measured from what the discussion proposes as a biological explanation.

6. التفسير البيولوجي طبقة مستقلة

7. A defensible final report

الجين ذو الدلالة الإحصائية ليس تلقائيًا هدفًا بيولوجيًا مهمًا. يجب أن يدمج التفسير حجم التأثير وعدم اليقين وجودة التعليق الوظيفي وسياق المسارات والأدلة السابقة وتصميم التجربة والتحقق المستقل. وينبغي للتقرير النهائي أن يميز بوضوح بين ما قاسه التحليل وما تقترحه المناقشة كتفسير بيولوجي.

Language / نوع الكود: **Code / Command**

1. Research question
2. Experimental design and sample metadata
3. Data source and accession
4. Raw-data QC
5. Preprocessing decisions and rationale
6. Quantification / alignment method
7. Statistical model and design formula

8. Multiple-testing correction

9. Effect sizes and uncertainty

10. Sensitivity analyses

11. Biological interpretation

7. تقرير نهائي يمكن الدفاع عنه

Language / نوع الكود: **Text / Report Skeleton**

The report is deliberately longer than a simple list of significant genes. A defensible analysis makes its assumptions visible, records decisions, and separates evidence from interpretation.

8. What success means in the final edition

التقرير أطول عمدًا من مجرد قائمة بالجينات ذات الدلالة. فالتحليل القابل للدفاع يجعل افتراضاته مرئية، ويسجل القرارات، ويفصل بين الدليل والتفسير.

Success is not the number of pages or the number of code blocks. It is whether the book can carry a learner from a first Python expression to a defensible analysis in which the data, model, workflow, software environment, security controls, and scientific claims are connected. Synthetic exercises make the first steps safe; real data, statistical models, workflow systems, and verification make the later steps honest.

8. ما معنى النجاح في الطبعة النهائية؟

Part 16 — Deep Practice and Verification

لا يقاس النجاح بعدد الصفحات أو عدد كتل الكود. بل بقدرة الكتاب على نقل المتعلم من أول تعبير في Python إلى تحليل يمكن الدفاع عنه، تكون فيه البيانات والنموذج وسير العمل وبيئة البرامج والضوابط الأمنية والادعاءات العلمية مترابطة. تجعل التمارين الاصطناعية الخطوات الأولى آمنة، بينما تجعل البيانات الحقيقية والنماذج الإحصائية وأنظمة سير العمل والتحقق الخطوات اللاحقة أكثر صدقًا.

Chapter 63 — Public Data Acquisition and Provenance

الفصل 63 — الحصول على البيانات العامة ومصدرها وتتبّعها

الباب 16 — الممارسة العميقة والتحقق

1. Why acquisition is part of analysis

Public data are not merely files behind a download link. Source identity, accession, reference assembly version, annotation release, download date, and a checksum are part of the identity of the material from which an analysis is built. If any of these change, a rerun may no longer reproduce the original analysis even when the code is unchanged.

1. لماذا يُعد الحصول على البيانات جزءًا من التحليل

2. Accession is not provenance by itself

البيانات العامة ليست مجرد ملف نضغط على رابط لتنزيله. هوية المصدر، ورقم الوصول، وإصدار التجميع المرجعي، ونسخة التعليق الجيني، وتاريخ التنزيل، وقيمة checksum كلها أجزاء من هوية المادة التي سيُبنى عليها الاستنتاج. إذا تغيّر أحد هذه العناصر فقد تصبح إعادة التحليل غير مطابقة للتحليل الأول حتى لو بقي الكود نفسه دون تغيير.

An accession identifies a database resource, but it does not by itself describe how the resource was used. A defensible record states what was downloaded, from which record, under which version, what transformations were applied, and which files resulted. A useful manifest therefore links each file to its source and to the workflow step that produced it.

2. رقم الوصول وحده لا يساوي مصدرًا قابلًا للتتبع

3. A provenance-first manifest

رقم الوصول يحدد موردًا في قاعدة بيانات، لكنه لا يصف وحده كيف استُخدم المورد. يجب تسجيل ما الذي نُزّل، ومن أي سجل، وبأي إصدار، وما التحويلات التي أجريت عليه، وما الملفات الناتجة. لذلك فإن manifest الجيد لا يحتفظ باسم الملف فقط، بل يربط الملف بمصدره وبالخطوة التي أنتجته.

A real project should begin with a small, explicit manifest that can be read without executing the pipeline. It should never invent unknown values. A missing value marked for verification is scientifically safer than a fabricated accession or reference release.

3. سجل مشروع يبدأ بالمصدر قبل التحليل

4. Immutable raw data and derived data

أفضل نقطة بداية للمشروع الواقعي هي سجل صغير واضح يمكن قراءته دون تشغيل البرنامج. هذا السجل لا يحاول تخمين معلومات غير معروفة؛ بل يضع القيم التي تحتاج إلى التحقق في مواضع ظاهرة، لأن اختلاق رقم وصول أو إصدار مرجعي أخطر من ترك قيمة غير مكتملة مع علامة واضحة تطلب التحقق.

Raw data should be treated as an immutable reference point. Trimming and filtering should not overwrite the raw file. Each transformation should create a

new artifact that can be linked to its input. Separating raw and derived data makes error diagnosis, reruns, and comparison between processing versions possible.

4. البيانات الخام غير القابلة للتعديل والبيانات المشتقة

5. Checksums and file identity

يجب التعامل مع البيانات الخام بوصفها نقطة مرجعية ثابتة. لا تُجرى عمليات trimming أو filtering فوق الملف الخام نفسه. تنتج كل خطوة ملفًا جديدًا يمكن ربطه بالمدخل الذي سبقه. هذا الفصل بين الخام والمشتق يجعل اكتشاف الخطأ وإعادة الخطوة ومقارنة نسختين من المعالجة أمرًا ممكنًا.

A SHA-256 checksum does not prove that a biological sample is valid or that an experiment was designed correctly. It identifies a particular file state. Recording and checking it after transfer helps establish computational file integrity, while biological validity remains a separate question.

5. Checksums وهوية الملف

6. Practical public-data acquisition pattern

قيمة SHA-256 لا تثبت صحة العينة البيولوجية ولا صحة التجربة، لكنها تثبت هوية الملف في لحظة معينة. لذلك يجب أن تُسجل مع الملف، وأن تُفحص بعد النقل أو النسخ إذا كانت سلامة الملف الحسابية مهمة. الفرق بين سلامة الملف وصحة الدليل البيولوجي يجب أن يبقى واضحًا.

The following pattern is intentionally a documented template rather than a fabricated accession. The placeholder should be replaced with a verified record before execution, and commands and outputs should be retained under provenance. The goal is to make the intended acquisition reconstructable rather than dependent on an undocumented link.

6. نمط عملي آمن للحصول على بيانات عامة

7. When metadata conflict with filenames

الأمر التالي يمثل نمطًا توثيقيًا لا رقم وصول حقيقيًا. يُستبدل العنصر النائب بسجل موثق قبل التنفيذ، ويُحفظ سجل الأوامر والمخرجات في مجلد provenance. الهدف هو أن يستطيع شخص آخر معرفة ما الذي كان من المفترض تنزيله، وليس أن يعتمد على رابط غامض محفوظ في ذاكرة الكاتب.

A filename is not the ultimate source of truth. If a filename says Control while the sample sheet says Treated, the pipeline should not silently choose one. The conflict should stop the workflow or become an explicitly documented review item. This simple rule prevents errors that can be syntactically valid yet biologically misleading.

7. عندما تتعارض البيانات الوصفية مع أسماء الملفات

8. Provenance as part of the scientific result

اسم الملف ليس مصدر الحقيقة النهائي. إذا قال اسم الملف إن العينة Control بينما يقول sample sheet إنها Treated، فلا ينبغي للكود أن يختار أحدهما بصمت. يجب إيقاف المسار أو تحويل التعارض إلى حالة تحتاج إلى مراجعة بشرية موثقة. هذه القاعدة البسيطة تمنع أخطاء تحليلية قد تبدو صحيحة نحويًا لكنها تقلب معنى المقارنة.

When a number or plot is presented without its data source, software versions, and comparison design, it becomes difficult to determine whether an observed difference is biological or computational. Provenance is therefore not administrative decoration; it is part of interpreting the result itself.

8. مصدر البيانات جزء من النتيجة العلمية

عندما يُعرض رقم أو رسم بياني دون مصدر البيانات وإصدار الأدوات وتصميم المقارنة، يصبح من الصعب تحديد ما إذا كان الاختلاف ناتجًا عن البيولوجيا أو عن طريقة المعالجة. لذلك فإن provenance ليس ملحقًا إداريًا في نهاية المشروع؛ إنه جزء من تفسير النتيجة نفسها.

```
from dataclasses import dataclass
from pathlib import Path
import hashlib
@dataclass(frozen=True)
class DataArtifact:
    path: Path
    accession: str
    sha256: str
    def calculate_sha256(self) -> str:
        h = hashlib.sha256()
        with self.path.open("rb") as handle:
            for chunk in iter(lambda: handle.read(1024 * 1024), b""):
                h.update(chunk)
        return h.hexdigest()
    def verify(self) -> bool:
        return self.calculate_sha256() == self.sha256
artifact = DataArtifact(
    path=Path("data/raw/sample_R1.fastq.gz"),
    accession="VERIFY_BEFORE_USE",
    sha256="RECORD_FROM_SOURCE"
)
if artifact.sha256 != "RECORD_FROM_SOURCE":
    raise ValueError("Missing verified checksum")
print(artifact.path, artifact.accession)
```

This block belongs to the realistic teaching workflow. Syntactic validity is not enough; inputs and outputs must be checked and tied to the data, references, and software version used.

هذه الكتلة جزء من التطبيق التعليمي الواقعي. لا تكفي سلامة الصياغة؛ يجب فحص المدخلات والمخرجات، وربطها بالبيانات والمراجع والنسخة البرمجية المستخدمة.

Expected and failure cases should be tested, the execution environment should be recorded, and accession numbers or results must never be fabricated. If an external tool is not installed in the execution environment, the code should be identified as environment-dependent rather than claimed to have been run.

Chapter 64 — RNA-seq Diagnostics, Designs, and Effect Interpretation

الفصل 64 — تشخيص RNA-seq وتصميماته وتفسير التأثيرات

يجب اختبار الحالات المتوقعة وحالات الفشل، وتسجيل بيئة التنفيذ، وعدم اختلاق أرقام وصول أو نتائج. إذا كانت الأداة الخارجية غير مثبتة في البيئة، يجب وصف الكود بأنه يتطلب تلك البيئة بدل الادعاء بأنه شُغّل.

1. Counts are observations, not continuous measurements

Gene-level read counts are count observations. Their statistical structure should not be reduced to a mean and standard deviation followed by an automatic continuous-data test. RNA-seq variability depends on sequencing depth, dispersion, and design structure, so differential analysis requires a model appropriate to the data.

1. العدّات ملاحظات وليست قياسات مستمرة عادية

2. The Negative Binomial model as the central bridge

عدد القراءات المنسوبة إلى جين هو قيمة عدّ، ولا ينبغي اختزال طبيعتها الإحصائية في متوسط وانحراف معياري ثم تطبيق اختبار مناسب لبيانات مستمرة بصورة آلية. التباين في بيانات RNA-seq يتأثر بعمق القراءة وبالتشتت وبنية التصميم، ولذلك تحتاج المقارنة إلى نموذج يلائم طبيعة البيانات.

Modern differential-expression workflows use negative-binomial models to represent counts with overdispersion that a simple Poisson model cannot capture. The key idea is not memorizing an equation but understanding that the model separates mean-dependent variation from between-sample dispersion and uses the experimental design to estimate effects.

2. النموذج السالب ذي الحدين بوصفه الجسر الإحصائي

3. Dispersion is not a nuisance number

في التحليل التفاضلي الحديث تُستخدم نماذج سالبة ذات حدين لتمثيل العدّ مع التشتت الزائد الذي لا يلتقطه Poisson البسيط. الفكرة المهمة ليست حفظ المعادلة، بل فهم أن النموذج يميز بين التباين المتوقع من حجم العدّ وبين التشتت الذي يظهر بين العينات، ثم يستخدم تصميم التجربة لتقدير التأثيرات المطلوبة.

Dispersion represents additional variability among samples beyond what a simpler count model would expect. Its estimation directly affects uncertainty and statistical evidence. Reading a DESeq2 or edgeR result as only a table of p-values hides the model that produced the table.

3. التشتت ليس رقمًا هامشيًا

4. Design formula before code

التشتت يعبّر عن مقدار الاختلاف الإضافي بين العينات مقارنة بما يتوقعه نموذج عدّي أبسط. تقديره يؤثر مباشرة في عدم اليقين وفي قوة الدليل الإحصائي. لذلك فإن قراءة مخرجات DESeq2 أو edgeR دون فهم التشتت تجعل الجدول النهائي يبدو كأنه مجموعة من قيم p فقط، بينما النموذج الذي أنتجها أغنى بكثير.

Before writing analysis code, state the experimental question: which factor is being tested, are batches present, are observations paired or repeated, and is an interaction of interest? A design formula is not a programming detail; it is the translation of the scientific question into a statistical model.

4. صيغة التصميم قبل الكود

5. Why a t-test can still be educational but cannot be the endpoint

قبل كتابة أي سطر للتحليل يجب صياغة السؤال التجريبي: ما العامل الذي نريد تقدير أثره؟ هل توجد دفعات؟ هل توجد أزواج أو عينات متكررة؟ هل نريد اختبار تفاعل بين عاملين؟ اختيار design formula ليس تفصيلًا برمجيًا؛ إنه ترجمة السؤال العلمي إلى نموذج.

A t-test on a small teaching example can illustrate null hypotheses, differences between means, and uncertainty. It should not be the endpoint for real RNA-seq analysis. The learner should move from this intuition to count models, dispersion, GLMs, design formulas, and multiple-testing correction.

5. لماذا يمكن أن يبقى اختبار t تعليميًا لكنه لا يصلح كنهاية

6. Reading log2 fold change correctly

يمكن لاختبار t على مثال صغير أن يوضح فكرة الفرضية الصفرية والفرق بين المتوسطات وعدم اليقين. لكنه لا يمثل النموذج القياسي الذي يجب أن يتوقف عنده تحليل RNA-seq الحقيقي. ينبغي أن ينتقل المتعلم بعد هذا المثال إلى العدّ، والتشتت، وGLM، والتصميم، والتصحيح عن الاختبارات المتعددة.

Log2 fold change describes the direction and magnitude of an effect on a logarithmic scale. Its sign depends on the contrast and level ordering, while its magnitude describes effect size. It should never be interpreted without checking the contrast, and effect size should not be equated with statistical significance.

6. قراءة log2 fold change بصورة صحيحة

7. A realistic DESeq2 skeleton

قيمة log2 fold change تصف اتجاه وحجم التأثير على مقياس لوغاريتمي. الإشارة تحدد اتجاه المقارنة وفق ترتيب المستويات، بينما القيمة المطلقة تعطي مقياسًا لحجم الفرق. لا يجوز تفسيرها دون معرفة contrast المستخدم، ولا ينبغي مساواة حجم التأثير بالدلالة الإحصائية.

The following skeleton shows the structure of a real DESeq2 analysis without fabricating a count matrix or numerical result. File names and versions must be pinned in a real project. The important lesson is the explicit connection among metadata, count data, design, and contrast.

7. هيكل واقعي لتحليل DESeq2

8. Diagnostics before interpretation

المثال التالي يوضح البنية الواقعية للتحليل دون اختلاق مصفوفة عدّ أو نتائج. أسماء الملفات والإصدارات يجب تثبيتها من مشروع حقيقي. الفائدة التعليمية هنا أن الكود يربط metadata بمصفوفة العد وبصيغة التصميم ثم يطلب contrast صريحًا.

Before reading a gene list, inspect sample structure. PCA, sample relationships, quality distributions, batch structure, and MA plots are diagnostic tools rather than decoration. They help reveal whether the model is facing expected structure or a problem that must be understood before biological interpretation.

8. التشخيص قبل التفسير

قبل قراءة قائمة الجينات يجب فحص بنية العينات. PCA، والعلاقات بين العينات، وتوزيع الجودة، والدفعات، والـMA plot ليست زينة. إنها أدوات تكشف ما إذا كان النموذج يواجه بنية بيانات متوقعة أو مشكلة تحتاج إلى تفسير قبل الانتقال إلى الاستنتاج البيولوجي.

Language / نوع الكود: **R / Bioconductor**

```
library(DESeq2)
# counts: integer count matrix; columns must match metadata row names
# coldata: sample metadata with condition and any required batch terms
coldata$condition <- factor(coldata$condition)
```

```
dds <- DESeqDataSetFromMatrix(
countData = counts,
colData = coldata,
design = ~ batch + condition
)
dds <- DESeq(dds)
res <- results(
dds,
contrast = c("condition", "treated", "control")
)
res <- res[order(res$padj, na.last = NA), ]
head(res)
```

Chapter 65 — Workflow Engineering at Scale

الفصل 65 — هندسة سير العمل على نطاق واسع

1. From scripts to workflows

When an analysis becomes a chain of many steps, scattered Bash scripts are no longer sufficient. Dependencies need to be explicit: which file produces which output, what can run in parallel, and what must be rerun after failure. This is where a workflow manager becomes valuable.

1. من السكربتات المنفردة إلى سير العمل

2. Snakemake mental model

عندما يصبح التحليل سلسلة من عشرات الخطوات، لا يعود وجود ملفات Bash متفرقة كافيًا. نحتاج إلى وصف صريح للاعتماديات: أي ملف ينتج أي ملف، وما الذي يمكن تشغيله بالتوازي، وما الذي يجب إعادة تشغيله بعد الفشل. هنا يظهر دور مدير سير العمل.

Snakemake expresses workflows through rules that connect inputs and outputs. If one file changes, the system can determine the affected part of the dependency graph rather than forcing a manual rerun of every step. This is fundamentally different from a linear command list whose dependencies exist only in the author’s memory.

2. النموذج الذهني لـSnakemake

3. Nextflow mental model

الفكرة الأساسية في Snakemake هي تعريف rules تربط المدخلات بالمخرجات. لا يحتاج المستخدم إلى إعادة تشغيل كل شيء يدويًا إذا تغيّر ملف واحد؛ يمكن للنظام تحديد الجزء المتأثر من الرسم البياني. هذا يختلف جذريًا عن قائمة أوامر متسلسلة لا تعرف اعتمادياتها إلا من ترتيبها.

Nextflow uses processes and channels to express data flow, while portability across environments becomes part of the workflow design. The important lesson is not memorizing syntax but designing a pipeline that can be observed, rerun, scaled, and documented.

3. النموذج الذهني لـNextflow

4. Environment isolation

في Nextflow تصبح القنوات والعمليات ومصادر البيانات جزءًا من وصف سير العمل، وتصبح قابلية النقل عبر البيئات جزءًا من التصميم. القيمة الحقيقية ليست حفظ syntax بعينه، بل فهم أن pipeline يجب أن يكون قابلًا للمراقبة، وإعادة التشغيل، والتوسع، والتوثيق.

A requirements.txt file is useful but does not always describe the operating system, native libraries, or external tools. Conda can manage package environments, while containers such as Docker or Apptainer provide a stronger isolation layer. The appropriate level depends on project requirements rather than fashion.

4. عزل البيئة البرمجية

5. A workflow should fail visibly

وجود requirements.txt مفيد لكنه لا يصف دائمًا نظام التشغيل أو المكتبات الأصلية أو الأدوات الخارجية. Conda يمكن أن يدير بيئة الحزم، بينما الحاويات مثل Docker أو Apptainer تضيف طبقة عزل أقوى. يجب اختيار مستوى العزل وفق متطلبات المشروع، لا وفق الموضة.

Silent failure is more dangerous than explicit failure. If a file is missing or a program fails but the pipeline continues and produces a final report, the existence of that report does not prove success. Exit codes, expected outputs, logs, and validation steps belong inside the pipeline.

5. يجب أن يفشل سير العمل بصورة ظاهرة

6. OOP for workflow state

الخطأ الصامت أخطر من الخطأ الصريح. إذا غاب ملف أو فشل برنامج ثم استمر المسار وأنتج تقريرًا نهائيًا، فإن وجود التقرير لا يعني نجاح التحليل. لذلك يجب أن تُعامل رموز الخروج، والملفات المتوقعة، والسجلات، وعمليات التحقق كجزء من pipeline نفسه.

OOP can be useful here when it represents real entities. A WorkflowRun object can carry a run identifier, version, inputs, outputs, and state. The goal is not to turn every line into a class, but to give real system entities explicit invariants and validation rules.

6. استخدام OOP لتمثيل حالة سير العمل

7. Reproducibility metadata

يمكن استخدام البرمجة الكائنية هنا بطريقة مفيدة: كائن WorkflowRun يحمل معرف التشغيل، والإصدار، والمدخلات، والمخرجات، والحالة. هذا لا يعني تحويل كل شيء إلى classes بلا حاجة؛ بل يعني أن الكائن يمثل كيانًا حقيقيًا في النظام وله قواعد تحقق واضحة.

Important workflow runs should leave a trace: workflow version, container or environment identity, inputs, checksums, execution time, resources, and success or failure state. This makes differences between runs inspectable instead of dependent on memory.

7. البيانات الوصفية لإعادة الإنتاج

8. A minimal Nextflow example

كل تشغيل مهم ينبغي أن يترك أثرًا: نسخة workflow، نسخة الحاوية أو البيئة، المدخلات، checksums، تاريخ التشغيل، الموارد المستخدمة، وحالة النجاح أو الفشل. هذه البيانات تجعل الفرق بين تشغيلين قابلًا للتحليل بدل الاعتماد على الذاكرة.

The following is not a complete sensitive-data pipeline; it illustrates the relationship between a process, its input, and its output. A real project should add versions, containers, reports, and validation rules.

8. مثال Nextflow صغير لكنه واقعي

المثال التالي ليس pipeline كاملًا لبيانات حساسة؛ إنه يوضح علاقة process بالمدخل والمخرج. في المشروع الحقيقي يجب إضافة الإصدارات، الحاويات، التقارير، وقواعد التحقق.

```
from dataclasses import dataclass, field
from pathlib import Path
from enum import Enum
class RunState(str, Enum):
    PENDING = "pending"
    RUNNING = "running"
    SUCCEEDED = "succeeded"
    FAILED = "failed"
@dataclass
class WorkflowRun:
```

run_id: str

```
    workflow_version: str
    inputs: list[Path] = field(default_factory=list)
    outputs: list[Path] = field(default_factory=list)
    state: RunState = RunState.PENDING
```

```
def validate_inputs(self) -> None:
    missing = [p for p in self.inputs if not p.exists()]
    if missing:
        raise FileNotFoundError(f"Missing inputs: {missing}")
def mark_success(self) -> None:
    if not self.outputs or not all(p.exists() for p in self.outputs):
        raise RuntimeError("Cannot mark run successful: outputs are missing")
    self.state = RunState.SUCCEEDED
run = WorkflowRun(
  run_id="2026-08-12-example",
  workflow_version="1.0.0",
  inputs=[Path("data/raw/sample.fastq.gz")],
  outputs=[Path("results/qc/sample_fastqc.html")],
)
run.validate_inputs()
```

Chapter 66 — AI-Assisted Analysis, Validation, and Scientific Judgment

الفصل 66 — التحليل بمساعدة الذكاء الاصطناعي والتحقق والحكم العلمي

1. AI is an assistant, not an evidence source

A generative model can accelerate code writing, test design, and reformulation of a research question. It does not become a source of scientific evidence merely because its answer sounds plausible. Claims must trace back to data, primary literature, or authoritative documentation.

1. الذكاء الاصطناعي مساعد وليس مصدرًا للدليل

2. AI-generated code needs a verification ladder

يمكن للنموذج التوليدي أن يسرّع كتابة الكود، أو اقتراح اختبارات، أو إعادة صياغة سؤال بحثي، لكنه لا يصبح مصدرًا للنتيجة العلمية لمجرد أنه أنتج نصًا مقنعًا. يجب أن يعود الادعاء العلمي إلى البيانات أو الورقة الأصلية أو التوثيق الرسمي.

Verification starts with syntax and moves through unit tests, boundary cases, expected outputs, and finally scientific appropriateness. Successful execution is not the end of validation; it is the beginning.

2. الكود المولّد بالذكاء الاصطناعي يحتاج إلى سلم تحقق

3. AI-assisted literature review without fabricated citations

التحقق يبدأ من syntax ثم ينتقل إلى الاختبار الوحدوي، وحالات الحدود، ومقارنة المخرجات المتوقعة، ثم التحقق من أن الطريقة نفسها مناسبة للسؤال البيولوجي. نجاح الكود في التنفيذ ليس نهاية التحقق بل بدايته.

AI can help turn a question into search terms, cluster papers, and draft comparison tables. Every citation must still be opened and verified and linked to the claim it supports. DOI, PMID, journal, and publication year are verification anchors, not decorative tokens appended to generated prose.

3. مراجعة الأدبيات بمساعدة الذكاء الاصطناعي دون مراجع مختلقة

4. AI-assisted data interpretation

يمكن استخدام AI لتحويل السؤال إلى كلمات بحث، وتصنيف الأوراق، وإنشاء مسودة جدول مقارنة. لكن كل مرجع يجب فتحه والتحقق من بياناته وربطه بالادعاء الذي يدعمه. DOI وPMID واسم المجلة وسنة النشر نقاط تحقق، وليست زينة توضع بعد إجابة مولدة.

AI can help explain a result table or suggest visualization strategies, but interpretation must start from the actual variables. A model should not be given context-poor data and then treated as an oracle for causal stories. Each interpretation should distinguish measured quantities, inferred conclusions, and remaining hypotheses.

4. تفسير البيانات بمساعدة الذكاء الاصطناعي

5. Privacy and governance

يمكن للنظام أن يساعد في شرح جدول نتائج أو اقتراح طرق لتصور البيانات، لكن التفسير يجب أن يبدأ من المتغيرات الفعلية. لا ينبغي إعطاء النموذج بيانات غير مفهومة السياق ثم قبول قصة سببية منها. كل تفسير يجب أن يوضح ما قيس، وما استُنتج، وما الذي بقي فرضية.

Before sending biological data to an external service, determine what the data are and what usage constraints apply. Clinical, proprietary, or restricted data do not become safe merely because a tool is popular. Depending on governance requirements, local models, approved institutional services, or de-identified data may be more appropriate.

5. الخصوصية والحوكمة

6. AlphaFold as a case study in responsible AI use

قبل إرسال بيانات بيولوجية إلى خدمة خارجية يجب معرفة طبيعة البيانات وشروط استخدامها. البيانات السريرية أو الملكية أو ذات القيود الخاصة لا تصبح آمنة لمجرد أن الأداة شائعة. في بعض السياقات يكون النموذج المحلي أو الخدمة المؤسسية المعتمدة أو البيانات المنزوعة الهوية خيارًا أفضل.

Structure-prediction systems illustrate how AI can be practically useful in bioinformatics. A structural prediction is not an experimental measurement of protein function. Confidence measures, model scope, prior knowledge, and independent evidence must be considered before making a strong biological claim.

6. AlphaFold بوصفه مثالًا على الاستخدام المسؤول للذكاء الاصطناعي

7. An auditable AI review record

توضح أدوات التنبؤ البنيوي كيف يمكن للذكاء الاصطناعي أن يغير المعلوماتية الحيوية عمليًا. لكن التنبؤ البنيوي ليس قياسًا تجريبيًا لوظيفة البروتين. يجب قراءة درجات الثقة ومجال النموذج، وربط التنبؤ بالمعرفة السابقة والأدلة المستقلة قبل صياغة استنتاج قوي.

When AI is used in a scientific project, record the task, model or service, date, non-sensitive inputs, human modifications, tests performed, and sources verified. This does not make the AI authoritative; it makes its contribution auditable.

7. سجل قابل للتدقيق لاستخدام الذكاء الاصطناعي

8. The final scientific judgment remains human and evidence-based

إذا استُخدم AI في مشروع علمي، فمن المفيد تسجيل المهمة، والنموذج أو الخدمة، والتاريخ، والمدخلات غير الحساسة، والتعديلات البشرية، والاختبارات التي أُجريت، والمصادر التي تم التحقق منها. هذا لا يحول الذكاء الاصطناعي إلى سلطة، بل يجعل أثره في العمل قابلًا للمراجعة.

The goal of AI in this book is not to replace the researcher or programmer. It is to reduce repetitive work and create more room for checking and reasoning. As tools become better at producing code and prose, the ability to recognize persuasive errors becomes more important, not less.

8. يبقى الحكم العلمي النهائي إنسانيًا ومبنيًا على الدليل

9. A practical AI-assisted review record

الهدف من AI في هذا الكتاب ليس استبدال الباحث أو المبرمج، بل إزالة الأعمال المتكررة وفتح مساحة أكبر للفحص والتفكير. وكلما زادت قدرة الأداة على إنتاج الكود والنصوص، زادت أهمية معرفة حدودها والقدرة على كشف الخطأ المقنع.

The following record shows a minimal project-level log. It contains no secrets or sensitive biological data and can be stored as YAML or an internal table.

9. سجل عملي لمراجعة العمل بمساعدة الذكاء الاصطناعي

السجل التالي يوضح الحد الأدنى الذي يمكن الاحتفاظ به مع المشروع. لا يحتوي على أسرار أو بيانات حساسة، ويمكن نسخه إلى ملف YAML أو جدول داخلي.

Language / نوع الكود: **YAML**

```yaml
ai_review:
task: "code_review_and_literature_triage"
service: "record approved model or institutional service"
date: "record execution date"
sensitive_data_shared: false
human_changes_recorded: true
tests_run:
- syntax
- unit
- expected_output
- boundary_cases
sources_verified: true
final_scientific_decision: "human_review_required"
```

Conclusion — A brief summary of the main conclusions, with reference to implications and potential next steps.

References — A list of all sources cited in the report.

Language / نوع الكود: **Text / IMRAD Structure**

```
هيكل التقرير العلمي (IMRAD):

I = Introduction (المقدمة)
M = Methods (المنهج)
R = Results (النتائج)
A = And
D = Discussion (المناقشة)
```

3. Writing a Clear Scientific Question

The scientific question is the core of the report. Everything that follows aims to answer this question. A good question should be:

3. كتابة سؤال علمي واضح

السؤال العلمي هو محور التقرير. كل ما يلي يهدف إلى الإجابة عن هذا السؤال. السؤال الجيد يجب أن يكون:

· محدداً (Specific): يتناول موضوعاً محدداً بوضوح.

· قابلاً للقياس (Measurable): يمكن الإجابة عنه باستخدام البيانات المتاحة.

· Specific: addresses a clearly defined topic.

· قابلاً للاختبار (Testable): يمكن التحقق منه من خلال التجربة أو التحليل.

· Measurable: can be answered using available data.

· ذا صلة (Relevant): مهم في السياق العلمي.

· Testable: can be verified through experiment or analysis.

· Relevant: important in the scientific context.

مثال:

سؤال ضعيف: "ما هو دور الجينات في السرطان؟"

سؤال جيد: "ما هي الجينات ذات التعبير التفاضلي بين خلايا سرطان الثدي والخلايا السليمة في بيانات RNA-seq، وكيف ترتبط هذه الجينات بمسارات الإشارات المعروفة؟"

In the report, the question should be clearly stated in the introduction, and revisited in the discussion and conclusion.

4. Methods: What Did You Do and Why?

في التقرير، يجب أن يُذكر السؤال بوضوح في المقدمة، وأن يُعاد النظر إليه في المناقشة والاستنتاج.

The methods section is where you explain exactly what you did. It should be detailed enough for reproducibility, and should justify the project's choices. Key elements in the methods section:

4. المنهج: ما الذي فعلته ولماذا؟

قسم المنهج هو المكان الذي تشرح فيه بالضبط ما فعلته. يجب أن يكون مفصلاً بما يكفي لإعادة الإنتاج، ويجب أن يبرر اختياراتك. العناصر الأساسية في قسم المنهج:

· البيانات: مصدر البيانات، وصفها، وكيفية الحصول عليها. إذا كانت البيانات حية، سجل الإصدار وتاريخ الوصول. إذا كانت اصطناعية، صف كيف أُنشئت.

· الأدوات البرمجية: سجل أسماء الأدوات وإصداراتها (مثل Python 3.11، Biopython 1.81، R 4.3، DESeq2 1.40).

· Data: data source, description, and how it was obtained. If the data is live, record the version and access date. If synthetic, describe how it was created.

· المعاملات: سجل جميع المعاملات المستخدمة (مثل عتبة الجودة، عتبة التعبير التفاضلي، طريقة التطبيع، مستوى الدلالة).

· Software: record tool names and versions (e.g., Python 3.11, Biopython 1.81, R 4.3, DESeq2 1.40).

· الخطوات التحليلية: صف بالترتيب كل خطوة قمت بها، مع الإشارة إلى أي قرارات منهجية مهمة (مثل طريقة التعامل مع القيم المفقودة، أو اختيار نموذج إحصائي معين).

· Parameters: record all parameters used (e.g., quality threshold, differential expression threshold, normalization method, significance level).

· التبرير: اشرح لماذا اخترت هذه الأدوات والمعاملات وليس غيرها.

· Analytical steps: describe in order each step you performed, noting any important methodological decisions (e.g., how to handle missing values, or choosing a specific statistical model).

· Justification: explain why you chose these tools and parameters over others.

```
# (مثال: وصف منهجي في تقرير (نص وصفي، ليس كوداً تنفيذياً
"""
```

تم تطبيع بيانات التعبير باستخدام طريقة median-of-ratios من حزمة DESeq2 (الإصدار 1.40.0)،

لأن هذه الطريقة مناسبة لبيانات التعبير التفاضلي وتفترض أن معظم الجينات لا تتغير بشكل كبير.

تم تحديد الجينات ذات التعبير التفاضلي باستخدام عتبة FDR < 0.05 و |log2FC| > 1.

```
"""
```

5. Results: What Did You Find؟

5. النتائج: ما الذي وجدته؟

The results section is the objective core of the report. Here you present what you found without interpretation. Use tables and figures to present data in an organized way. Results should be:

قسم النتائج هو القلب الموضوعي للتقرير. هنا تعرض ما وجدته دون تفسير. استخدم الجداول والرسوم البيانية لعرض البيانات بشكل منظم. يجب أن تكون النتائج:

· موضوعية: تقدم الحقائق كما هي، دون انحياز.

· Objective: present facts as they are, without bias.

· منظمة: تُعرض بترتيب منطقي، غالباً ما يتبع ترتيب الأسئلة الفرعية أو خطوات التحليل.

· Organized: presented in a logical order, often following the order of sub-questions or analysis steps.

· مدعومة بالبيانات: كل ادعاء في النتائج يجب أن يكون مدعوماً ببيانات معروضة في جدول أو رسم.

· Data-supported: every claim in the results should be supported by data shown in a table or figure.

· قابلة للتفسير: حتى بدون المناقشة، يجب أن تكون النتائج مفهومة.

· Interpretable: even without the discussion, the results should be understandable.

مبدأ: فصل النتائج عن التفسير.

النتائج: "أظهر تحليل التعبير التفاضلي 45 جيناً مختلفاً بين المجموعتين (FDR < 0.05)."

التفسير (في المناقشة): "هذه الجينات قد تلعب دوراً في تطور المرض، حيث تشير الأدبيات إلى أن العديد منها مرتبط بالالتهاب والاستجابة المناعية."

6. Discussion: What Do the Results Mean؟

6. المناقشة: ماذا تعني النتائج؟

The discussion section is where you interpret the results in the broader context. Here you connect what you found to what is known, derive meaning, and raise new questions. A good discussion:

قسم المناقشة هو المكان الذي تفسر فيه النتائج في السياق الأوسع. هنا تربط ما وجدته بما هو معروف، وتستخلص المعنى، وتطرح الأسئلة الجديدة. المناقشة الجيدة:

· تفسر النتائج: لا تعيد سرد النتائج، بل تشرح معناها.

· Interprets the results: does not restate results, but explains their meaning.

· تقارن مع الدراسات السابقة: كيف تتفق أو تختلف نتائجك مع ما هو معروف؟

· Compares with previous studies: how do the project's results agree or disagree with what is known?

· تناقش القيود: ما هي حدود تحليلك، وكيف تؤثر على الاستنتاجات؟

· Discusses limitations: what are the limits of the project's analysis, and how do they affect conclusions?

· تطرح أسئلة جديدة: ما هي الخطوات التالية؟ ما الذي يمكن فعله لتعزيز النتائج أو اختبار الفرضيات بشكل أعمق؟

· Raises new questions: what are the next steps? What can be done to strengthen the results or test hypotheses more deeply?

مبدأ: المناقشة هي المكان الذي تنتقل فيه من البيانات إلى المعنى.

استخدمها لبناء قصة علمية متماسكة، دون تجاوز حدود الأدلة.

7. Limitations and Uncertainty: Scientific Honesty

Documenting limitations is not an admission of weakness, but an essential part of scientific integrity. Every study has limits, and acknowledging them increases the project's credibility. In the limitations section (or within the discussion), mention:

7. القيود وعدم اليقين: الصدق العلمي

توثيق القيود ليس اعترافاً بالضعف، بل هو جزء أساسي من النزاهة العلمية. كل دراسة لها حدود، والاعتراف بها يزيد

من مصداقيتك. في قسم القيود (أو ضمن المناقشة)، اذكر:

· حجم العينة: هل العينة كافية للاستدلال المطلوب؟

· جودة البيانات: هل هناك مشاكل في جودة القراءات، أو التلوث، أو التحيزات التقنية؟

· Sample size: is the sample sufficient for the desired inference?

· الافتراضات الإحصائية: هل تم التحقق من افتراضات النماذج الإحصائية؟ هل هي معقولة؟

· Data quality: are there issues with read quality, contamination, or technical biases?

· التعميم: هل يمكن تعميم النتائج على مجموعات سكانية أو ظروف أخرى؟

· Statistical assumptions: were the assumptions of statistical models checked? Are they reasonable?

· العوامل غير المقاسة: هل هناك متغيرات مربكة لم تُقَيّد؟

· Generalizability: can the results be generalized to other populations or conditions?

· Unmeasured factors: are there confounding variables that were not controlled?

مثال على توثيق القيود:

"تفسير النتائج يجب أن يأخذ في الاعتبار أن حجم العينة محدود (12 عينة فقط)، وأن تأثيرات الدفعات قد تكون موجودة رغم محاولة ضبطها. كما أن النتائج تعتمد على بيانات RNA-seq، التي تقيس مستويات الـ RNA وليس البروتين، لذا قد لا تعكس النشاط البروتيني الفعلي. هناك حاجة إلى تحقق تجريبي مستقل لتأكيد هذه النتائج."

8. Citation and Disclosure

Citation is an essential part of the scientific report. You must cite all sources you relied on, whether data, tools, or ideas. Good citation:

8. الاستشهاد والإفصاح

الاستشهاد هو جزء أساسي من التقرير العلمي. يجب أن تستشهد بكل المصادر التي اعتمدت عليها، سواء كانت بيانات، أو أدوات، أو أفكاراً. الاستشهاد الجيد:

· Gives credit: acknowledges others' contributions.

· يمنح الفضل: يعترف بمساهمات الآخرين.

· Supports arguments: shows the project's conclusions are built on prior knowledge.

· يدعم الحجج: يظهر أن استنتاجاتك مبنية على المعرفة السابقة.

· Facilitates verification: allows verification against the original sources.

· يسهل التحقق: يسمح للقراء بالرجوع إلى المصادر الأصلية.

مبادئ الاستشهاد:

1. استشهد بالمصدر الأولي كلما أمكن (وليس بالمراجعة التي تذكره).

2. استخدم معرفات دائمة (DOI، PMID) لتحديد المصادر.

3. اتبع أسلوب الاستشهاد المطلوب من المجلة أو المؤسسة.

Disclosure means revealing any potential conflicts of interest, such as funding, commercial relationships, or affiliations that might affect the objectivity of the report. Transparency in disclosure builds trust in the scientific work.

الإفصاح (Disclosure) يعني الكشف عن أي تضارب محتمل في المصالح، مثل التمويل، أو العلاقات التجارية، أو الانتماءات التي قد تؤثر على موضوعية التقرير. الشفافية في الإفصاح تعزز الثقة في العمل العلمي.

9. Practical Application: Writing a Report for the Brick by Brick Project

9. التطبيق العملي: كتابة تقرير لمشروع Brick by Brick

Let us apply these principles to the Brick by Brick project you completed in the previous chapter. This is an example of the report structure:

لنطبق هذه المبادئ على مشروع Brick by Brick الذي أنجزته في الفصل السابق. هذا مثال لهيكل التقرير:

```
# تقرير مشروع Brick by Brick: تحليل بيانات RNA-seq اصطناعية
##الملخص
```

استخدم هذا المشروع بيانات RNA-seq اصطناعية لتحديد الجينات ذات التعبير التفاضلي بين عينات سليمة ومريضة. تم تطبيق سير عمل متكامل باستخدام Bash، Biopython، وR. أظهرت النتائج 45 جيناً مختلفاً إحصائياً، مع رسوم بيانية تلخص التوزيع والتجميع. أظهر تحليل PCA تجمعاً واضحاً للعينات حسب الحالة. النتائج تؤكد صلاحية سير العمل وتوضح أهمية الفصل بين النتائج الحسابية والتفسير البيولوجي.

```
## 1. المقدمة
```

يهدف هذا المشروع إلى تطبيق أدوات المعلوماتية الحيوية في سير عمل متكامل، باستخدام بيانات اصطناعية، للإجابة عن السؤال: ما هي الجينات ذات التعبير التفاضلي بين العينات السليمة والمريضة؟ يساعد هذا المشروع في فهم كيفية ترابط مراحل التحليل من البيانات الخام إلى التقرير النهائي.

```
## 2. المنهج
```

تم توليد بيانات اصطناعية: 100 جين، 12 عينة (6 سليمة، 6 مريضة)، مع 20 جيناً مختلفة بين المجموعتين. استخدمت Python مع Biopython لتحليل التسلسلات، وR مع DESeq2 لتحليل التعبير التفاضلي. تم تطبيق فحوصات أمنية دفاعية على التسلسلات، مع تطبيق مبدأ fail-closed. تم توثيق جميع الإصدارات والمعاملات.

```
## 3. النتائج
```

أظهر تحليل التعبير التفاضلي 45 جيناً مختلفاً ($FDR < 0.05$، $|log2FC| > 1$). رسوم PCA أظهرت تجمع العينات حسب الحالة، مع وجود تأثير دفعة طفيف. رسوم Volcano و MA أظهرت توزيع الجينات. أظهر تحليل التسلسلات تنوعاً في الطول ونسبة GC.

```
## 4. المناقشة
```

النتائج تتوافق مع التصميم الاصطناعي للبيانات، حيث تم توليد 20 جيناً مختلفاً، واكتشف التحليل 45 جيناً (بما في ذلك بعض الإيجابيات الكاذبة المتوقعة). التجمع في PCA يؤكد أن الحالة هي المصدر الرئيسي

للتباين. تأثير الدفعة الملاحظ يشير إلى أهمية تسجيل البيانات الوصفية. القيود تشمل حجم العينة الصغير واستخدام بيانات اصطناعية، مما يستدعي التحقق على بيانات حقيقية.

6. المراجع

1. Cock, P. J. A. et al. (2009). Biopython: freely available Python tools... Bioinformatics, 25(11), 1422-1423.

2. Love, M. I. et al. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15(12), 550.

3. Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer.

5. الاستنتاج

هذا المشروع أظهر جدوى سير العمل المتكامل في تحليل بيانات RNA-seq، مع التأكيد على أهمية التوثيق، والفحوصات الأمنية، والفصل بين النتائج والتفسير. يمكن تطبيق هذا الإطار على بيانات حقيقية بعد التحقق من صحة الأدوات والمعاملات.

10. Limits of the Scientific Report

The scientific report is a powerful tool, but it has limits:

10. حدود التقرير العلمي

التقرير العلمي أداة قوية، لكن له حدود:

· لا يعوض عن تصميم سيء: إذا كان التصميم التجريبي أو التحليلي معيباً، فلن يصلحه التقرير.

· It does not compensate for poor design: if the experimental or analytical design is flawed, the report will not fix it.

· يعتمد على جودة التوثيق: التقرير الجيد يتطلب توثيقاً دقيقاً لكل خطوة.

· It depends on documentation quality: a good report requires accurate documentation of every step.

· قد يكون متحيزاً: حتى مع النوايا الحسنة، قد يختار الباحث عرض نتائج معينة وإغفال أخرى. المراجعة من قبل الأقران تساعد في تقليل هذا التحيز.

· It may be biased: even with good intentions, researchers may choose to present some results and omit others. Peer review helps reduce this bias.

· لا يثبت السببية: التقرير يقدم أدلة، لكن الإثبات النهائي للسببية قد يتطلب تجارب إضافية وتكراراً.

· It does not prove causation: the report provides evidence, but final proof of causation may require additional experiments and replication.

قاعدة: التقرير العلمي هو نقطة البداية، وليس النهاية. هو أداة للتواصل، وليس حقيقة مطلقة.

11. Best Practices for Writing the Scientific Report

To ensure the quality of the project’s report, follow these practices:

11. ممارسات أفضل لكتابة التقرير العلمي

لضمان جودة تقريرك، اتبع هذه الممارسات:

· اكتب بوضوح وإيجاز: استخدم لغة مباشرة، وتجنب المصطلحات غير الضرورية.

· استخدم الرسوم البيانية والجداول: وسائل بصرية تعزز الفهم وتلخص البيانات الكبيرة.

· راجع تقريرك: اقرأه عدة مرات، واطلب من زميل مراجعته.

· تحقق من الاستشهادات: تأكد من دقة جميع الاستشهادات، واستخدم معرفات دائمة.

· Write clearly and concisely: use direct language, avoid unnecessary jargon.

· كن صادقاً بشأن القيود: لا تحاول إخفاء نقاط الضعف؛ اعترف بها ووضح كيف أثرت على النتائج.

· Use figures and tables: visual aids enhance understanding and summarize large data.

· افصل بين النتائج والتفسير: النتائج هي ما وجدته، والتفسير هو ما تعتقده.

· Review the project’s report: read it multiple times, and ask a colleague to review it.

· اتبع تعليمات المجلة: إذا كنت تنوي النشر، اتبع تنسيق المجلة المستهدفة.

· Check citations: ensure all citations are accurate, and use persistent identifiers.

· Be honest about limitations: do not try to hide weaknesses; acknowledge them and explain how they affected results.

· Separate results from interpretation: results are what you found, interpretation is what you think it means.

· Follow journal guidelines: if you intend to publish, follow the target journal's format.

The scientific report is the first step toward publication. After writing the report, the workflow can submit it to a scientific journal, or to the project’s supervisor, or to the project’s team. Publication adds a layer of peer review, enhancing the work's credibility. But even if not published, a good report is a valuable document that documents the project’s work and facilitates communication with others.

التقرير العلمي هو الخطوة الأولى نحو النشر. بعد كتابة التقرير، يمكن تقديمه إلى مجلة علمية، أو إلى مشرفك، أو إلى فريقك. النشر يضيف طبقة من المراجعة من قبل الأقران، مما يعزز مصداقية العمل. لكن حتى لو لم تُنشر، التقرير الجيد هو وثيقة قيمة توثق عملك وتسهل التواصل مع الآخرين.

خمسة أسئلة تضبط التفكير العلمي

1. Central idea and its limits — Explain the central idea of "The Scientific Report" without overstating its evidence

1. الفكرة المركزية وحدودها — شرح الفكرة المركزية في "التقرير العلمي" دون تجاوز حدودها العلمية

Example of overstatement: claiming that "a good scientific report is one that presents only exciting results." This is incorrect; a good report presents balanced evidence, including negative results and limitations. The correct statement is: "a good scientific report is a document that presents a clear question, appropriate methods, documented results, and an honest discussion of limitations and uncertainty, separating computational evidence from biological interpretation." A simple test: Practical check: "Does my report mention limitations and clarify what the results did not prove, or does it focus only on positive aspects?".

مثال على التجاوز: القول إن "التقرير العلمي الجيد هو الذي يقدم نتائج مثيرة فقط". هذا غير صحيح؛ التقرير الجيد يقدم أدلة متوازنة، بما في ذلك النتائج السلبية والقيود. الصحيح هو: "التقرير العلمي الجيد هو وثيقة تقدم سؤالاً واضحاً، ومنهجاً مناسباً، ونتائج موثقة، ومناقشة صريحة للقيود وعدم اليقين، وتفصل بين الأدلة الحسابية والتفسير البيولوجي." اختبار بسيط: فحص تطبيقي: "هل تقريري يذكر القيود ويوضح ما الذي لم تثبته النتائج، أم أنه يركز فقط على الجوانب الإيجابية؟".

Example: The report states that "RNA-seq analysis showed 45 differentially expressed genes between the groups" – this is a computational description. Claiming that "these genes cause the disease" is a biological claim that goes beyond what the computational results prove. A good report separates results (what the analysis showed) from discussion (what these results might mean in biological context). Practical test: before writing a causal claim in the project’s report, Practical check: "Do these results prove causation, or are they just a correlation or statistical difference that needs further verification?".

مثال: التقرير يذكر أن "تحليل RNA-seq أظهر 45 جيناً مختلفاً بين المجموعتين" هذا وصف حسابي. الادعاء بأن "هذه الجينات تسبب المرض" هو ادعاء بيولوجي يتجاوز ما تثبته النتائج الحسابية. التقرير الجيد يفصل بين النتائج (ما أظهره التحليل) والمناقشة (ما قد تعنيه هذه النتائج في السياق البيولوجي). اختبار عملي: قبل أن تكتب ادعاءً سببيّاً

في تقريرك، فحص تطبيقي: "هل هذه النتائج تثبت السببية، أم أنها مجرد ارتباط أو اختلاف إحصائي يحتاج إلى تحقق إضافي؟"ـ

مثال تطبيقي على تقرير علمي لمشروع تحليل RNA-seq:

· المدخلات: ملفات FASTQ الخام، جينوم مرجعي (hg38)، بيانات وصفية عن العينات، معاملات التحليل (عتبة الجودة، عتبة التعبير التفاضلي)ـ

· الافتراضات: أن القراءات تمثل العينة بدقة، وأن الجينوم المرجعي مناسب، وأن طريقة التطبيع والنموذج الإحصائي مناسبان للبياناتـ

· المخرجات: قائمة الجينات ذات التعبير التفاضلي، رسوم بيانية (Volcano، MA، PCA)، تقارير جودة، سجلات التشغيلـ

· القيود: حجم العينة محدود (n=12)، قد تكون هناك تأثيرات دفعات غير مقيّدة، النتائج تحتاج إلى تحقق تجريبي مستقل، التحليل يعتمد على جودة القراءات وكفاءة المحاذاةـ

Applied example of a scientific report for an RNA-seq analysis project:

اختبار: قبل كتابة التقرير، تأكد من أنك قد حددت جميع المدخلات والافتراضات والمخرجات والقيود بوضوح. إذا لم تستطع تحديد أحدها، فالتقرير غير مكتملـ

· Inputs: raw FASTQ files, reference genome (hg38), sample metadata, analysis parameters (quality threshold, differential expression threshold).

· Assumptions: that the reads accurately represent the sample, that the reference genome is appropriate, and that the normalization method and statistical model are suitable for the data.

· Outputs: list of differentially expressed genes, plots (Volcano, MA, PCA), quality reports, and logs.

· Limitations: limited sample size (n=12), potential unmeasured batch effects, results require independent experimental verification, analysis depends on read quality and alignment efficiency.

Test: before writing the report, ensure that the workflow has clearly identified all inputs, assumptions, outputs, and limitations. If you cannot identify one of them, the report is incomplete.

عند كتابة التقرير، يجب تسجيل:

· المصدر: البيانات والمجلات والمراجع المستخدمةـ

· الإصدار: إصدارات البرمجيات وقواعد البيانات.

· المعلمات: جميع المعاملات والعتبات المستخدمة.

· التحويلات: خطوات المعالجة بالترتيب.

When writing the report, record:

اختبار: فحص تطبيقي: "لو أعطيت هذا التقرير (مع السجلات والملفات) لزميل بعد عام، هل يستطيع فهم ما فعلته وإعادة تشغيل التحليل دون مساعدتي؟" إذا كانت الإجابة لا، فالتوثيق غير كافٍ.

· Version: software and database versions.

· Parameters: all parameters and thresholds used.

· Transformations: processing steps in order.

Test: Practical check: "If I gave this report (with logs and files) to a colleague a year later, could they understand what I did and rerun the analysis without my help?" If the answer is no, the documentation is insufficient.

في تقريرك، وضح أن المشروع استخدم بيانات اصطناعية، وهذا يسمح باختبار سير العمل قبل الانتقال إلى بيانات حقيقية. اشرح كيف أن استخدام البيانات الاصطناعية ساعد في التحقق من صحة الكود والمعاملات، وكشف الأخطاء المحتملة. إذا كان المشروع ينتقل إلى بيانات حقيقية، اذكر أن النتائج على البيانات الاصطناعية لا تغني عن التحقق على البيانات الحقيقية، وأن أي استنتاجات بيولوجية تتطلب تأكيداً مستقلاً.

In the project’s report, clarify that the project used synthetic data, which allows testing the workflow before moving to real data. Explain how using synthetic data helped verify the code and parameters, and detect potential errors. If the project moves to real data, mention that results on synthetic data do not replace verification on real data, and that any biological conclusions require independent confirmation.

اختبار عملي: في تقريرك، اكتب فقرة تشرح كيف أن البيانات الاصطناعية استخدمت لاختبار سير العمل، وما هي الدروس المستفادة من هذا الاختبار قبل الانتقال إلى البيانات الحقيقية (إن وجدت).

Practical test: in the project’s report, write a paragraph explaining how synthetic data was used to test the workflow, and what lessons were learned from this test before moving to real data (if applicable).

Conclusion

الخاتمة

The book closes on the same distinction that governs its technical work: computation can represent, transform, compare, validate, and report biological data, but a computational result is not automatically a biological proof. The strongest workflow is the one that makes its inputs, assumptions, transformations, software versions, and limitations visible.

تختتم الكتابة بالفكرة التي تحكم العمل التقني كله: تستطيع الحوسبة أن تمثل البيانات البيولوجية وتحولها وتقارنها وتتحقق منها وتعرضها، لكن النتيجة الحسابية لا تصبح إثباتًا بيولوجيًا لمجرد أن البرنامج عمل دون خطأ. أقوى سير عمل هو الذي يجعل مدخلاته وافتراضاته وتحويلاته وإصدارات برامجه وحدوده ظاهرة وقابلة للفحص.

The progression from Biopython objects and object-oriented design to statistical analysis, reproducible workflows, and real-data practice is therefore not a collection of unrelated tools. Each layer adds a new kind of control: representation controls meaning, tests control behavior, provenance controls traceability, statistical models control inference, and workflow systems control reproducibility at scale.

ولهذا فإن الانتقال من كائنات Biopython والبرمجة الكائنية إلى التحليل الإحصائي وسير العمل القابل لإعادة الإنتاج والبيانات الحقيقية ليس مجموعة أدوات منفصلة. كل طبقة تضيف نوعًا جديدًا من الضبط: التمثيل يضبط المعنى، والاختبارات تضبط السلوك، ومصدر البيانات يضبط قابلية التتبع، والنماذج الإحصائية تضبط الاستدلال، وأنظمة سير العمل تضبط إعادة الإنتاج على نطاق واسع.

The advanced chapters also close the gap between controlled educational data and real sequencing practice. Real FASTQ files can contain adapter remnants, quality variation, duplication, compositional bias, batch effects, and metadata problems. These are not exceptions to the workflow; they are part of the evidence that must be diagnosed before interpretation.

وتغلق الفصول المتقدمة كذلك الفجوة بين البيانات التعليمية المضبوطة والممارسة الفعلية مع بيانات التسلسل. فقد تحتوي ملفات FASTQ الحقيقية على بقايا المحولات، وتفاوت في الجودة، وتكرارات، وانحيازات تركيبية، وتأثيرات للدفعات، ومشكلات في البيانات الوصفية. وهذه ليست تفاصيل هامشية؛ بل جزء من الدليل الذي يجب تشخيصه قبل تفسير النتائج.

For RNA-seq, the conceptual bridge from simple group comparison to negative-binomial count models, generalized linear models, dispersion estimation, and multiple-testing control is essential. A pedagogical t-test can introduce the logic of comparison, but the inferential engine must match the structure of sequencing counts and the experimental design.

وفي RNA-seq، فإن الجسر المفاهيمي من مقارنة المجموعات البسيطة إلى نماذج العد السالب ذي الحدين والنماذج الخطية العامة وتقدير التشتت وضبط المقارنات المتعددة أمر أساسي. قد يشرح اختبار t التعليمي منطق المقارنة، لكن أداة الاستدلال يجب أن تتوافق مع طبيعة عدادات التسلسل وتصميم التجربة.

The security material deliberately remains bounded. Educational fail-closed checks and synthetic fixtures are useful for learning defensive design, but they are not substitutes for operational governance, threat modeling, validated reference data, access control, audit trails, independent review, and institutional policy. The boundary is part of the scientific result, not a disclaimer added after it.

وتظل مادة الأمن الحيوي محددة الحدود عمدًا. فاختبارات الفشل الآمن والبيانات الاصطناعية التعليمية مفيدة لتعلم التصميم الدفاعي، لكنها لا تحل محل الحوكمة التشغيلية ونمذجة التهديدات والبيانات المرجعية الموثقة وضبط الوصول وسجلات التدقيق والمراجعة المستقلة والسياسات المؤسسية. وهذه الحدود جزء من النتيجة العلمية نفسها، وليست تحذيرًا يضاف بعد انتهاء العمل.

Finally, modern practice requires reproducible workflow managers, isolated environments, and disciplined use of generative AI. Snakemake, Nextflow, Conda, Docker, Apptainer, and AI-assisted tools can accelerate work, but acceleration does not remove the need for validation, provenance, human judgment, or independent evidence. The durable skill is not memorizing one tool; it is constructing an auditable chain from question to defensible conclusion.

وأخيرًا، تتطلب الممارسة الحديثة أنظمة إدارة سير عمل قابلة لإعادة الإنتاج وبيئات معزولة واستخدامًا منضبطًا للذكاء الاصطناعي التوليدي. يمكن لـ Snakemake وNextflow وConda وDocker وApptainer وأدوات المساعدة بالذكاء الاصطناعي أن تسرّع العمل، لكن السرعة لا تلغي الحاجة إلى التحقق ومصدر البيانات والحكم البشري والأدلة المستقلة. والمهارة الباقية ليست حفظ أداة واحدة، بل بناء سلسلة قابلة للتدقيق من السؤال العلمي إلى استنتاج يمكن الدفاع عنه.

Reference Appendices

Appendix A — Educational FASTA Files — الملحق أ — ملفات FASTA التعليمية

Small synthetic FASTA files can be bundled so that exercises work without an internet connection. External datasets should retain their identity, version, and creation or download date.

تُستخدم ملفات FASTA اصطناعية صغيرة حتى تعمل التمارين دون اتصال بالإنترنت. يجب توثيق هوية الملف وإصداره وتاريخ إنشائه أو تنزيله عندما تكون البيانات خارجية.

Appendix B — Unit Tests — الملحق ب — اختبارات الوحدة

Safe tests can cover complement and reverse_complement, FASTA parsing, rejection of invalid symbols, and the gene_drive_lock gate in dry-run mode.

تشمل الاختبارات الآمنة التحقق من complement وreverse_complement، وقراءة FASTA، ورفض الرموز غير الصالحة، واختبار بوابة gene_drive_lock في وضع dry-run.

Appendix C — biobrick_security Package Structure — الملحق ج — بنية حزمة biobrick_security

Policy, analysis, and audit logging should be separated, and the package should be divided into small testable modules.

تُفصل السياسة عن التحليل وسجل التدقيق، وتُقسم الحزمة إلى وحدات صغيرة قابلة للاختبار.

```
biobrick_security/
├── __init__.py
├── screening.py
├── off_target.py
├── policy.py
├── watermark.py
├── models.py
├── audit.py
└── tests/
```

A public installation command or publication claim should not appear before a real repository, successful tests, and a documented release exist. Distinguishing “can be published” from “has actually been published” is part of scientific honesty.

لا يُكتب أمر تثبيت أو ادعاء بالنشر العام قبل وجود مستودع حقيقي، واختبارات ناجحة، وإصدار موثق. الفرق بين «يمكن نشره» و«نُشر فعلًا» جزء من الأمانة العلمية.

Matching count-matrix columns to metadata rows is essential. If the mapping is wrong, counts may be assigned to the wrong samples without a syntax error; the explicit check is part of analytical integrity.

مختبر تطبيقي — تطبيق واقعي قابل للاختبار

تطابق أعمدة مصفوفة العد مع صفوف metadata شرط أساسي. إذا كان الربط خاطئًا فقد تنتقل counts إلى العينة الخطأ دون خطأ نحوي؛ لذلك stopifnot جزء من سلامة التحليل.

```
library(DESeq2)
counts <- read.csv("results/gene_counts.csv",
                   row.names = 1,
                   check.names = FALSE)
meta <- read.csv("metadata/samples.csv",
                 row.names = 1)
stopifnot(identical(colnames(counts), rownames(meta)))
meta$condition <- factor(meta$condition,
                         levels = c("control", "treated"))
meta$batch <- factor(meta$batch)
dds <- DESeqDataSetFromMatrix(
countData = round(as.matrix(counts)),
colData = meta,
design = ~ batch + condition
)
dds <- dds[rowSums(counts(dds) >= 10) >= 2, ]
dds <- DESeq(dds)
print(resultsNames(dds))
res <- results(dds,
        contrast = c("condition", "treated", "control"))
```

Explicit resources, inputs, and outputs make the process portable. A command that works on a laptop is not yet an operational workflow; its execution requirements must be explicit.

```
nextflow.enable.dsl=2
```

تعريف الموارد والمدخلات والمخرجات يجعل العملية قابلة للنقل إلى بيئات مختلفة. لا يكفي نجاح الأمر على الحاسوب الشخصي؛ يجب أن تكون متطلبات التنفيذ صريحة.

process FASTQC {
tag "${sample_id}"
cpus 4
memory "4 GB"

```
input:
  tuple val(sample_id), path(read1)
output:
  path "${sample_id}_fastqc.html"
  path "${sample_id}_fastqc.zip"
```

script:
"""
fastqc --threads ${task.cpus} ${read1}
"""
}

في سير العمل، يجب قراءة العلاقة بين المدخلات والمخرجات والموارد والسجل باعتبارها رسمًا سببيًا للتحليل. نجاح خطوة منفردة لا يثبت أن الخط كله قابل لإعادة الإنتاج؛ الإصدار والبيئة والبيانات المرجعية جزء من التعريف الكامل للعملية.

The rule binds a sample to explicit input, output, and log artifacts. rule all declares the desired end state, while fastqc defines how it is produced. Production runs should pin the environment and record parameters.

rule all:
input:
expand("qc/{sample}_R1_fastqc.html", sample=["S01", "S02"])

القاعدة تربط العينة بمدخلها ومخرجاتها وسجلها. rule all يحدد النتيجة النهائية، وfastqc يصف طريقة إنتاجها. في التشغيل الحقيقي يجب تثبيت البيئة وتسجيل المعاملات.

rule fastqc:
input:
"data/raw/{sample}_R1.fastq.gz"
output:
html="qc/{sample}_R1_fastqc.html",
zip="qc/{sample}_R1_fastqc.zip"
log:
"logs/fastqc/{sample}.log"
threads: 4
shell:
"fastqc --threads {threads} --outdir qc {input} > {log} 2>&1"

A checksum does not prove biological correctness, but it identifies the exact byte content processed. If it changes between stages, comparability should not be assumed until the difference is understood.

```
import hashlib
from pathlib import Path
```

الـchecksum لا يثبت أن الملف صحيح بيولوجيًا، لكنه يثبت هوية البايتات التي عولجت. إذا اختلف checksum بين مرحلتين فلا ينبغي افتراض قابلية المقارنة حتى يُعرف السبب.

```
def sha256_file(path: Path, chunk_size: int = 1024 * 1024) -> str:
    digest = hashlib.sha256()
    with path.open("rb") as handle:
        for chunk in iter(lambda: handle.read(chunk_size), b""):
            digest.update(chunk)
    return digest.hexdigest()
path = Path("data/raw/sample_R1.fastq.gz")
print(path.name, sha256_file(path))
```

Repeated FASTQ identifiers are not direct proof of PCR duplicates. The metric is deliberately named duplicate_ids; PCR or optical duplication requires broader tooling and context.

```
from dataclasses import dataclass
```

عدّ معرفات مكررة في FASTQ ليس إثباتًا مباشرًا لـ PCR duplicates. المتغير يسمى duplicate_ids عمدًا؛ أما تقدير PCR أو optical duplicates فيحتاج أدوات وسياقًا أوسع.

```
@dataclass
class ReadQC:
    records: int = 0
    total_bases: int = 0
    gc_bases: int = 0
    duplicate_ids: int = 0
def add(self, identifier: str, sequence: str, seen: set[str]) -> None:
        self.records += 1
        self.total_bases += len(sequence)
        self.gc_bases += sum(base in "GCgc" for base in sequence)
        if identifier in seen:
            self.duplicate_ids += 1
        seen.add(identifier)
def as_dict(self) -> dict:
        return {
            "records": self.records,
            "bases": self.total_bases,
            "gc_fraction": self.gc_bases / self.total_bases if self.total_bases else 0,
            "duplicate_ids": self.duplicate_ids,
        }
```

Sample-sheet validation is scientific, not administrative. A duplicated identifier, missing batch, or malformed replicate can change the meaning of the statistical model, so the program should fail early and explicitly.

```
from pathlib import Path
import csv
```

التحقق من جدول العينات خطوة علمية وليست إدارية. خطأ في اسم العينة أو تكرار معرف أو غياب batch قد يغير معنى النموذج الإحصائي كله، ولذلك يفشل البرنامج مبكرًا برسالة واضحة.

```
def read_sample_sheet(path: Path) -> list[Sample]:
    required = {"sample_id", "condition", "batch", "replicate", "fastq_r1"}
    samples = []
with path.open(newline="", encoding="utf-8") as handle:
        reader = csv.DictReader(handle)
        columns = set(reader.fieldnames or [])
        if not required.issubset(columns):
            raise ValueError(f"Missing columns: {sorted(required - columns)}")
seen = set()
        for row in reader:
            sample = Sample(
                sample_id=row["sample_id"],
                condition=row["condition"],
                batch=row["batch"],
                replicate=int(row["replicate"]),
                fastq_r1=row["fastq_r1"],
                fastq_r2=row.get("fastq_r2") or None,
            )
            sample.validate()
            if sample.sample_id in seen:
                raise ValueError(f"Duplicate sample_id: {sample.sample_id}")
            seen.add(sample.sample_id)
            samples.append(sample)
if not samples:
        raise ValueError("Sample sheet is empty")
    return samples
```

The Sample object represents experimental assumptions. Validation prevents an invalid sample from silently entering analysis, while manifest_row turns that state into an auditable record.

```
from dataclasses import dataclass
from datetime import datetime
```

كائن Sample يمثل افتراضات التصميم التجريبي. وجود validate يمنع وصول عينة غير صالحة إلى التحليل، بينما manifest_row يحول هذه الحالة إلى سجل يمكن توثيقه.

```
@dataclass(frozen=True)
class Sample:
    sample_id: str
    condition: str
    batch: str
    replicate: int
    fastq_r1: str
    fastq_r2: str | None = None
def validate(self) -> None:
        if not self.sample_id.strip():
            raise ValueError("sample_id must not be empty")
        if self.condition not in {"control", "treated"}:
            raise ValueError("unknown condition")
        if self.replicate < 1:
            raise ValueError("replicate must be positive")
```

```
    if not self.fastq_r1.endswith((".fastq", ".fastq.gz")):
        raise ValueError("R1 must be FASTQ")
def manifest_row(self) -> dict:
    return {
        "sample_id": self.sample_id,
        "condition": self.condition,
        "batch": self.batch,
        "replicate": self.replicate,
        "fastq_r1": self.fastq_r1,
        "fastq_r2": self.fastq_r2,
        "validated_at": datetime.utcnow().isoformat(timespec="seconds"),
    }
sample = Sample("S01", "treated", "B01", 1, "data/raw/S01_R1.fastq.gz")
sample.validate()
print(sample.manifest_row())
```

QC is not the statistical model

وجود التحقق أو معالجة الأخطاء يحدد ما الذي يعد فشلًا صريحًا. هذه نقطة مهمة لأن الخطأ الصامت قد ينتج ملفًا يبدو صحيحًا نحويًا بينما يحمل عينة أو عمودًا أو قيمة في غير موضعها. في السياق الحيوي، المخرجات الحسابية هنا تمثل بيانات أو فرضيات قابلة للفحص، وليست دليلًا بيولوجيًا مستقلًا. أي نتيجة لاحقة يجب أن تُربط بمصدر البيانات، وتصميم التجربة، وإصدار الأدوات، وخطوات المعالجة. البرمجة الكائنية هنا ليست غاية أسلوبية؛ فائدتها أن القواعد والبيانات وحالة الكائن يمكن جمعها في بنية واحدة، ثم اختبارها قبل السماح لها بالمرور إلى مرحلة لاحقة من سير العمل.

Quality control asks whether the data and process are fit for analysis. The statistical model asks how much evidence supports a specified comparison under stated assumptions. Passing QC does not replace differential-expression modeling, and a favorable p-value does not repair broken metadata.

الفرق بين QC والتحليل الإحصائي

Reading RNA-seq results without reducing them to a p-value

Quality control يسأل: هل البيانات والعملية مناسبتان للتحليل؟ أما النموذج الإحصائي فيسأل: ما مقدار الدليل على الاختلاف تحت افتراضات محددة؟ لا يجوز استخدام نجاح PCA أو انخفاض نسبة القراءات الرديئة كبديل عن نموذج التعبير التفاضلي، كما لا يجوز استخدام قيمة p جيدة لإخفاء عيب واضح في metadata.

A p-value is not a complete description of a result. Effect size, uncertainty, multiplicity, data structure, and the experimental question all matter. A small log2 fold change can be statistically detectable in a large experiment, while a biologically important effect may remain uncertain in a small experiment. Results should therefore be read as a body of evidence rather than a single number.

قراءة نتائج RNA-seq دون الوقوع في فخ قيمة p

لا تكفي قيمة p وحدها لوصف أهمية النتيجة. يجب النظر إلى حجم التأثير، وعدم اليقين في تقديره، وعدد الاختبارات، وشكل البيانات، والفرضية التي صيغ التصميم من أجلها. الجين ذو log2 fold change صغير جدًا قد يكون ذا دلالة إحصائية في عينة كبيرة، بينما قد يكون التأثير البيولوجي المهم غير حاسم إحصائيًا في عينة صغيرة. لذلك ينبغي أن تُقرأ النتيجة كحزمة من الأدلة لا كرقم منفرد.

GitHub — What is GitHub Copilot? and official Copilot documentation. https://docs.github.com/en/copilot/

References and Sources

المراجع والمصادر

المراجع العلمية المحورية

1. Watson, J. D. و Crick, F. H. (1953). البنية الجزيئية للأحماض النووية: بنية لحمض الديوكسي ريبونوكلييك. Nature, 171, 737–738.

1. Watson, J. D., & Crick, F. H. (1953). Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. Nature, 171, 737–738. https://doi.org/10.1038/171737a0

2. Alberts, B. وآخرون (2022). البيولوجيا الجزيئية للخلية، الطبعة السابعة. W. W. Norton.

2. Alberts, B. et al. (2022). Molecular Biology of the Cell (7th ed.). W. W. Norton.

3. Lodish, H. وآخرون (2021). البيولوجيا الخلوية الجزيئية، الطبعة التاسعة. W. H. Freeman.

3. Lodish, H. et al. (2021). Molecular Cell Biology (9th ed.). W. H. Freeman.

4. Cock, P. J. A. وآخرون (2009). Biopython: أدوات Python متاحة بحرية للبيولوجيا الجزيئية الحاسوبية والمعلوماتية الحيوية. Bioinformatics, 25(11), 1422–1423.

4. Cock, P. J. A. et al. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

5. Love, M. I. وHuber, W. وAnders, S. (2014). التقدير المنضبط للتغير الطوي والتشتت لبيانات RNA-seq باستخدام DESeq2. Genome Biology, 15, 550.

5. Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15, 550.

6. Jinek, M. وآخرون (2012). إنزيم قطع DNA مبرمج وموجّه بواسطة RNA مزدوج في المناعة البكتيرية التكيفية. Science, 337, 816–821.

6. Jinek, M. et al. (2012). A programmable dual-RNA-guided DNA endonuclease in adaptive bacterial immunity. Science, 337, 816–821.

7. Doudna, J. A. وCharpentier, E. (2014). الجبهة الجديدة لهندسة الجينوم باستخدام CRISPR-Cas9. Science, 346, 1258096.

7. Doudna, J. A., & Charpentier, E. (2014). The new frontier of genome engineering with CRISPR-Cas9. Science, 346, 1258096.

8. Hsu, P. D. وLander, E. S. وZhang, F. (2014). تطوير CRISPR-Cas9 وتطبيقاته في هندسة الجينوم. Cell, 157, 1262–1278.

8. Hsu, P. D., Lander, E. S., & Zhang, F. (2014). Development and applications of CRISPR-Cas9 for genome engineering. Cell, 157, 1262–1278.

9. Wickham, H. (2016). ggplot2: الرسوم البيانية الأنيقة لتحليل البيانات. Springer.

9. Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer.

10. منظمة الصحة العالمية (2022). الإطار التوجيهي العالمي للاستخدام المسؤول لعلوم الحياة: الحد من المخاطر الحيوية وحوكمة أبحاث الاستخدام المزدوج.

10. World Health Organization (2022). Global Guidance Framework for the Responsible Use of the Life Sciences: Mitigating Biorisks and Governing Dual-Use Research.

11. Jumper, J وآخرون (2021). التنبؤ عالي الدقة ببنية البروتين باستخدام AlphaFold. Nature, 596, 583–589.

11. Jumper, J. et al. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596, 583–589.

12. Crick, F. (1970). العقيدة المركزية في البيولوجيا الجزيئية. Nature, 227, 561–563.

12. Crick, F. (1970). Central Dogma of Molecular Biology. Nature, 227, 561–563. https://doi.org/10.1038/227561a0

13. Anders, S وHuber, W. (2010). تحليل التعبير التفاضلي لبيانات عدّ التسلسلات. Genome Biology, 11, R106.

13. Anders, S., & Huber, W. (2010). Differential expression analysis for sequence count data. Genome Biology, 11, R106.

14. Robinson, M. D., McCarthy, D. J., & Smyth, G. K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics, 26(1), 139–140.

15. Li, H., Handsaker, B., Wysoker, A. et al. (2009). The Sequence Alignment/Map format and SAMtools. Bioinformatics, 25(16), 2078–2079.

16. Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics, 25(14), 1754–1760.

17. Ewels, P., Magnusson, M., Lundin, S., & Käller, M. (2016). MultiQC: summarize analysis results for multiple tools and samples in a single report. Bioinformatics, 32(19), 3047–3048.

18. Altschul, S. F. et al. (1990). Basic local alignment search tool. Journal of Molecular Biology, 215(3), 403–410.

19. Durbin, R., Eddy, S. R., Krogh, A., & Mitchison, G. (1998). Biological Sequence Analysis. Cambridge University Press.

20. Compeau, P. E. C., & Pevzner, P. A. (2018). Bioinformatics Algorithms: An Active Learning Approach (2nd ed.).

14. Robinson, M. D., McCarthy, D. J., & Smyth, G. K. (2010). edgeR: حزمة Bioconductor لتحليل التعبير التفاضلي لبيانات التعبير الجيني الرقمي. Bioinformatics, 26(1), 139–140.

15. Li, H., Handsaker, B., Wysoker, A., وآخرون (2009). صيغة Sequence Alignment/Map وأدوات SAMtools. Bioinformatics, 25(16), 2078–2079.

16. Li, H., & Durbin, R. (2009). محاذاة سريعة ودقيقة للقراءات القصيرة باستخدام تحويل Burrows–Wheeler. Bioinformatics, 25(14), 1754–1760.

17. Ewels, P., Magnusson, M., Lundin, S., & Käller, M. (2016). MultiQC: تلخيص نتائج التحليل لأدوات وعينات متعددة في تقرير واحد. Bioinformatics, 32(19), 3047–3048.

18. Altschul, S. F., وآخرون (1990). أداة البحث الأساسية للمحاذاة المحلية. Journal of Molecular Biology, 215(3), 403–410.

19. Durbin, R., Eddy, S. R., Krogh, A., & Mitchison, G. (1998). تحليل التسلسلات البيولوجية. Cambridge University Press.

20. Compeau, P. E. C., & Pevzner, P. A. (2018). خوارزميات المعلوماتية الحيوية: منهج التعلم النشط، الطبعة الثانية.

.Garland Science .الوراثة الجزيئية البشرية، الطبعة الخامسة .Read, A. P. (2018) و.Strachan, T .21

21. Strachan, T., & Read, A. P. (2018). Human Molecular Genetics (5th ed.). Garland Science.

.McGraw-Hill .علم الوراثة: التحليل والمبادئ، الطبعة السابعة .Brooker, R. J. (2020) .22

22. Brooker, R. J. (2020). Genetics: Analysis and Principles (7th ed.). McGraw-Hill.

Verhandlungen des naturforschenden .تجارب حول تهجين النباتات .Mendel, G. (1866) .23

.Vereines in Brünn, 4, 3–47

23. Mendel, G. (1866). Versuche über Pflanzen-Hybriden. Verhandlungen des naturforschenden Vereines in Brünn, 4, 3–47.

Official and Software Documentation

الوثائق الرسمية وتوثيق البرمجيات

توثيق Python — تثبيت Windows والبيئات.

Python Documentation — Windows installation and environment documentation. https://docs.python.org/3/using/windows.html

توثيق Biopython.

Biopython Documentation. https://biopython.org/docs/latest/

واجهة SeqRecord في Biopython الرسمية.

Biopython SeqRecord API. https://biopython.org/docs/latest/api/Bio.SeqRecord.html

NCBI Datasets.

NCBI Datasets. https://www.ncbi.nlm.nih.gov/datasets/

Bioconductor.

Bioconductor. https://www.bioconductor.org/

توثيق DESeq2.

DESeq2 documentation. https://bioconductor.org/packages/DESeq2/

توثيق edgeR.

edgeR documentation. https://bioconductor.org/packages/edgeR/

توثيق Snakemake.

Snakemake documentation. https://snakemake.github.io/

توثيق Nextflow.

Nextflow documentation. https://nextflow.io/

توثيق Apptainer.

Apptainer documentation. https://apptainer.org/

توثيق GitHub Copilot.

GitHub Copilot documentation. https://docs.github.com/en/copilot/

نظرة عامة على AlphaFold / DeepMind.

AlphaFold / DeepMind overview. https://deepmind.google/science/alphafold/